

VISIT...

LANZAROTE
Caliente.COM

Зализняк, Виктор Евгеньевич

Основы научных вычислений. Введение в численные методы для физиков: Учебное пособие для студентов естественно-научных и технических специальностей высших учебных заведе. - М.: Едиториал УРСС, 2002.- 296 с. : ил.
ISBN 5-354-00138-2, 800 экз.

Книга предназначена для использования в курсе численных методов. В ней рассматриваются такие вопросы, как решение уравнений, вычисление собственных значений и интегралов, интерполяция и аппроксимация функций, а также численное решение задачи Коши и краевой задачи для обыкновенных дифференциальных уравнений. Книга содержит множество примеров, демонстрирующих применение рассматриваемых методов. В дополнение приводится разнообразный справочный материал и краткий обзор библиотек программ, широко используемых в научных вычислениях. Для студентов естественно-научных и технических специальностей высших учебных заведений

Численные методы решения уравнений

ББК 22.193я73

Оглавление

ПРЕДИСЛОВИЕ - 9-10с.

ГЛАВА 0 Возникновение ошибок вычислений при выполнении арифметических операций на компьютере - 11-16с.

ГЛАВА 1 Решение уравнения $f(x)=0$ - 17-38с.

1.1 Метод деления отрезка пополам - 19-21с.

1.2 Вычисление корней с использованием итерационных функций - 21-37с.

1.2.1 Одноточечный итерационный процесс - 25-33с.

1.2.2 Многоточечный итерационный процесс - 33-35с.

1.2.3 Одноточечный итерационный процесс с памятью - 35-37с.

1.3 Заключительные замечания - 37-38с.

ГЛАВА 2 Решение систем линейных уравнений - 39-76с.

2.1 Необходимые сведения из линейной алгебры - 40-43с.

2.2 Системы линейных уравнений - 43-44с.

2.3 Типы матриц, часто встречающиеся при решении задач - 44-47с.

2.4 Источники ошибок - 47-48с.

2.5 Число обусловленности - 48-51с.

2.6 Прямые методы - 51-57с.

2.6.1 Основные принципы прямых методов - 52-55с.

2.6.2 Оценка ошибки приближенного решения - 55-56с.

2.6.3 Заключительные замечания - 56-57с.

2.7 Итерационные методы - 57-74с.

2.7.1 Основные принципы итерационных методов - 57-60с.

2.7.2 Метод Якоби - 60-63с.

2.7.3 Метод Гаусса-Зейделя - 63-64с.

2.7.4 Метод релаксации - 64-69с.

2.7.5 Вариационно-итерационные методы - 69-74с.

2.8 Какие методы более эффективны: прямые или итерационные? - 74-76с.

ГЛАВА 3 Вычисление собственных значений и векторов - 77-92с.

3.1 Основные сведения из линейной алгебры, относящиеся к задаче на собственное значение - 78-79с.

3.2 Локализация собственных значений - 79-80с.

3.3 Степенной метод - 80-83с.

3.4 Метод обратной итерации - 83-85с.

3.5 Итерации со сдвигом начала - 85-88с.

3.6 Применение ортогональных преобразований (QR-метод) - 88-91с.

3.7 Заключительные замечания - 91-92с.

ГЛАВА 4 Решение систем нелинейных уравнений - 93-112с.

4.1 Метод простой итерации - 94-100с.

4.2 Метод Ньютона - 100-105с.

4.3 Метод с кубической сходимостью - 105-107с.

4.4 Модификации метода Ньютона - 107-109с.

4.5 Повышение надёжности метода Ньютона - 109-112с.

ГЛАВА 5 Численное интегрирование - 113-144с.

5.1 Простейшие квадратурные формулы - 114-121с.

5.2 Вычисление интегралов с заданной точностью - 121-126с.

5.3 Формулы Гаусса-Кристоффеля - 126-132с.

5.4 Несобственные интегралы - 132-135с.

5.4.1 Бесконечные пределы интегрирования - 132-133с.

5.4.2 Подинтегральная функция имеет особенность - 133-135с.

5.5 Многомерное интегрирование - 135-144с.

ГЛАВА 6 Введение в конечно-разностные схемы для обыкновенных дифференциальных уравнений - 145-212с.

6.1 Простейший пример конечно-разностной схемы - 146-149с.

6.2 Определения аппроксимации и устойчивости - 149-160с.

6.2.1 Аппроксимация дифференциального уравнения разностной схемой - 150-155с.

6.2.2 Замена производных разностными отношениями - 155-158с.

6.2.3 Определение устойчивости разностной схемы - 158с.

6.2.4 Сходимость как следствие аппроксимации и устойчивости - 158-160с.

6.3 Численное решение задачи Коши - 160-189с.

6.3.1 Необходимое условие устойчивости разностных схем для линейных задач - 162-165с.

6.3.2 Методы Рунге-Кутты - 165-170с.

6.3.3 Методы Адамса - 170-175с.

6.3.4 Исследование устойчивости разностных схем в случае нелинейных задач - 175-177с.

6.3.5 Системы дифференциальных уравнений - 177-184с.

6.3.6 Методы решения жёстких систем дифференциальных уравнений - 184-189с.

6.4 Численное решение краевых задач - 189-206с.

6.4.1 Метод стрельбы - 189-192с.

6.4.2 Сведение разностной схемы к системе уравнений - 193-196с.

6.4.3 Метод последовательных приближений - 196-199с.

6.4.4 Метод установления - 199-203с.

6.4.5 Аппроксимация граничных условий в случае, когда на границе задано значение производной - 204-206с.

6.5 Оценка ошибки приближённого решения - 207-212с.

ГЛАВА 7 Интерполяция и приближение функций - 244с.

7.1 Интерполяция - 213-228с.

7.1.1 Интерполяционные полиномы Лагранжа и Ньютона - 211-218с.

7.1.2 Тригонометрические интерполяционные полиномы - 218-222с.

7.1.3 Сплайн интерполяция - 222-225с.

7.1.4 Двумерная интерполяция - 225-228с.

7.2 Приближение функций и представление данных - 228-244с.

7.2.1 Метод наименьших квадратов - 229-234с.

7.2.2 Приближение функции набором ортогональных функций - 234-241с.

7.2.3 Использование интерполяционных полиномов для приближения функций - 242-244с.

ПРИЛОЖЕНИЕ А Узлы и веса некоторых квадратурных формул Гаусса-Кристоффеля - 245-260с.

ПРИЛОЖЕНИЕ В Преобразование некоторых регулярных областей интегрирования в прямоугольные области - 261-262с.

ПРИЛОЖЕНИЕ С Условия устойчивости для некоторых разностных схем Рунге-Кутты, Адамса и предиктор-корректор - 263с.

ПРИЛОЖЕНИЕ D Краткий обзор программного обеспечения - 263-290с.

Библиотека подпрограмм LAPACK для задач линейной алгебры - 263-270с.

Библиотека подпрограмм IMSL® - 271-278с.

Библиотека подпрограмм Группы по численным алгоритмам (NAG®) - 278-288с.

Вычислительные функции среды MATLAB® - 288-290с.

ЛИТЕРАТУРА - 291-292с.

ПРЕДМЕТНЫЙ УКАЗАТЕЛЬ - 293-295с.

Предисловие

Развитие современной науки и технологии в значительной степени основано на компьютерном моделировании. Для того чтобы освоить основные принципы и методы компьютерного моделирования, студенты, прежде всего, должны овладеть основами классических численных методов, которые и составляют предмет этой книги. В настоящее время, различное программное обеспечение широко используется на практике, и невозможно грамотно и эффективно применять это обеспечение без глубокого понимания основ численных методик, на которых эти программы основаны. В этой книге, наряду с теоретическими аспектами вычислений, приводится краткий обзор библиотек программ, широко используемых в научных исследованиях.

Эта книга предназначена для использования в курсе численных методов для студентов физико-технических специальностей, а также она может быть полезна аспирантам в качестве справочника по численным методам. Объём этой книги рассчитан на изучение в течение одного семестра. Уровень изложения материала предполагает что читатель освоил курсы математического анализа (два семестра), линейной алгебры (один семестр) и дифференциальных уравнений (один семестр).

В небольшой по объёму книге невозможно рассмотреть все проблемы классического численного анализа. Поэтому, были выбраны наиболее важные, с моей точки зрения, темы, составляющие основу прикладных численных методов.

Виктор Зализняк, Мельбурн, октябрь 2001 г.

ГЛАВА 0

Возникновение ошибок вычислений при выполнении арифметических операций на компьютере

Компьютерная арифметика отличается от привычной нам арифметики. В традиционной математике, числа могут иметь бесконечное число цифр. В компьютерной арифметике, каждое допустимое число имеет конечное число цифр. Входные данные и результаты арифметических операций помещаются в оперативную память компьютера в определённой форме. Вещественные числа представляются в следующей стандартной форме (форма с плавающей точкой):

$$x = \pm b^c m_b \quad (0.1)$$

Здесь x – вещественное число, b есть основание системы счисления (обычно используются основания 2, 8 и 16), c называется характеристикой, m_b называется мантиссой (она определяет дробную часть числа), а $\pm$ обозначает знак числа x . Мантисса числа $x \neq 0$ удовлетворяет условию

$$\frac{1}{b} \leq m_b < 1,$$

что обеспечивает единственность представления (0.1). Любое вещественное число можно представить в форме с плавающей точкой и это может быть сделано в любой системе счисления с основанием $b > 0$. Однако, некоторые вещественные числа (например, иррациональные) невозможно записать в оперативную память компьютера точно. Все числа, которые могут быть помещены в оперативную память, представляются там в определённом виде. При этом их характеристики

ограничены: $c_m \leq c \leq c_p$, а мантиссы имеют вид конечных дробей по основанию b :

$$m_b = \frac{a_1}{b} + \frac{a_2}{b^2} + \dots + \frac{a_k}{b^k},$$

где коэффициенты a_n удовлетворяют условиям

$$1 \leq a_1 \leq b-1, 0 \leq a_n \leq b-1, n = 2, \dots, k.$$

Пример 0.1 (двоичное представление вещественного числа)

Рассмотрим вещественное число $x=8.75$. Представим это число в двоичной форме ($b=2$). Сначала мы переписем число x в виде $0.546875 \cdot 2^4$, тогда характеристика c , согласно (0.1), равна 4 (100 в двоичной форме). Дробная часть числа представляется в виде

$$0.546875 = \frac{1}{2} + \frac{1}{32} + \frac{1}{64},$$

тогда коэффициенты a_n равны

a_1	a_2	a_3	a_4	a_5	a_6	a_7	$\dots$	a_k
1	0	0	0	1	1	0	$\dots$	0

Для простоты, числа, которые могут быть помещены в оперативную память, мы будем называть машинными числами и они образуют конечное подмножество множества вещественных чисел. Для описания машинных чисел мы ввели параметры b , c_m , c_p и k , но использовать эти параметры на практике не очень удобно. Обычно, для описания машинных чисел вводится другой набор параметров: b , ε_0 , ε_1 , и ε_∞ . Эти параметры, как и предыдущие, зависят от конкретного компьютера и определяются следующим образом:

$$\varepsilon_0 = b^{c_m-1}$$

$$\varepsilon_1 = b^{1-k}$$

$$\varepsilon_\infty = b^{c_p}(1-b^{-k})$$

Эти параметры имеют следующие значения: ε_∞ - максимальное машинное число, ε_0 - минимальное машинное число и ε_1 есть минимальное расстояние между двумя машинными числами на интервале от 1 до b . Параметр ε_1 также называют относительной ошибкой определения единицы, так как все числа вида $1+x$ из интервала $[1, 1+\varepsilon_1]$ заменяются машинным числом 1 с ошибкой не превосходящей ε_1 . Следовательно, интервалы $(-\infty, -\varepsilon_\infty)$, $(-\varepsilon_0, 0)$, $(0, \varepsilon_0)$, $(1-\varepsilon_1/b, 1)$, $(1, 1+\varepsilon_1)$ и $(\varepsilon_\infty, \infty)$ не содержат машинных чисел. Наряду со стандартной точностью, существует возможность более точного представления машинных чисел. Это можно осуществить увеличив число цифр в мантиссе, и расширив область допустимых значений характеристик. Тогда машинные числа с повышенной точностью будут определяться некоторыми параметрами

$$b, c_m^*, c_p^*, k^*.$$

Например, определяющие параметры для РС представлены в Таблице 0.1.

Таблица 0.1

	Одинарная точность	Двойная точность
b	2	2
c_m	-125	-1021
c_p	128	1024
k	24	53
ε_0	2^{-126}	2^{-1022}
ε_1	2^{-23}	2^{-52}
ε_∞	$2^{128}(1-2^{-24})$	$2^{1024}(1-2^{-53})$

Если число $x \in [-\varepsilon_\infty, \varepsilon_\infty]$, то оно заменяется машинным числом x_m . В процессе такой замены используется какой-нибудь способ приближения (усечение или округление) и в качестве x_m выбирается ближайшее к x число. В результате такой замены возникает ошибка $x-x_m$. Для $\varepsilon_0 \leq |x| \leq \varepsilon_\infty$ эта ошибка оценивается величиной $\varepsilon_1 |x|$, то есть $|x-x_m| \leq \varepsilon_1 |x|$. Если $|x| \in (0, \varepsilon_0)$, тогда x_m может принимать одно из двух значений: 0 или ε_0 , и понятно что в обоих случаях $|x-x_m| \leq \varepsilon_0 |x|$. Поэтому параметр ε_0 также

называют абсолютной ошибкой определения нуля. Эти оценки можно объединить в следующее выражение:

$$x_m = x(1 + \alpha) + \beta, |\alpha| \leq \varepsilon_1, |\beta| \leq \varepsilon_0, \quad (0.2)$$

которое справедливо для всех $x \in [-\varepsilon_\infty, \varepsilon_\infty]$.

На компьютерах арифметические операции выполняются над машинными числами. Во многих случаях, результат этих операций может не быть машинным числом, или другими словами, результат арифметической операции (машинный результат), будучи машинным числом, может не совпадать с точным результатом. Например, для точного представления частного двух машинных чисел a и b часто требуется бесконечное количество цифр мантииссы. Поэтому, машинный результат отличается от точного результата деления a на b . Для дальнейшего анализа, мы введём следующее правило компьютерной арифметики: машинный результат арифметической операции над двумя машинными числами можно представить как результат записи точного значения этой операции в память компьютера. Для того чтобы отличать машинные операции от обычных (точных) операций, мы будем обозначать их таким же знаком в угловых скобках. Тогда, согласно принятому выше правилу, можно записать

$$\begin{aligned} a\langle + \rangle b &= (a + b)_m \\ a\langle - \rangle b &= (a - b)_m \\ a\langle \times \rangle b &= (a \times b)_m \\ a\langle / \rangle b &= (a / b)_m \end{aligned} \quad (0.3)$$

Формулы (0.3) имеют следующий смысл: машинный результат арифметической операции есть ближайшее к точному результату машинное число. Машинный результат также зависит от типа операции и чисел a и b . Например, если точный результат лежит вне интервала допустимых в компьютере чисел (результат не принадлежит $[-\varepsilon_\infty, \varepsilon_\infty]$), то мы сталкиваемся с ситуацией, называемой переполнением порядка. Эту ситуацию компьютер трактует как фатальную ошибку, и дальнейшие вычисления прекращаются. В правильно работающей программе такие ситуации должны быть исключены. Если абсолютное значение точного результата меньше чем ε_0 , то возникает ситуация,

называемая исчезновением порядка, и хотя это не является фатальной ошибкой такая ситуация также нежелательна.

Объединяя выражения (0.2) и (0.3), мы получим формулы для моделирования ошибок арифметических операций

$$a \langle + \rangle b = (a + b)_m = (a + b)(1 + \alpha) + \beta$$

$$a \langle - \rangle b = (a - b)_m = (a - b)(1 + \alpha) + \beta$$

$$a \langle \times \rangle b = (a \times b)_m = (ab)(1 + \alpha) + \beta$$

$$a \langle / \rangle b = (a / b)_m = (a / b)(1 + \alpha) + \beta$$

Эти формулы приводят к следующей оценке для ошибки возникающей при выполнении операций с плавающей точкой:

$$|a \langle * \rangle b - (a * b)| \leq \varepsilon_0 + \varepsilon_1 |a * b|,$$

где * обозначает любую из четырёх арифметических операций. Эта оценка может использоваться для анализа поведения ошибки при реализации различных вычислительных алгоритмов. Это очень важный вопрос, так как часто возникает ситуация, когда при выполнении некоторого вычислительного процесса происходит накопление ошибки, что приводит к значительному искажению конечного результата. Например, если предположить, что $\beta=0$, тогда ошибка вычисления

$$\prod_{n=1}^N a_n$$

может быть оценена как

$$\left| a_1 \langle \times \rangle a_2 \langle \times \rangle \dots \langle \times \rangle a_N - \prod_{n=1}^N a_n \right| \leq \varepsilon_1 (N-1) \prod_{n=1}^N |a_n|$$

ГЛАВА 1

Решение уравнения $f(x)=0$

Список обозначений

x^*	точное значение корня ($f(x^*)=0$)
I	интервал который содержит единственный корень
x_k	приближенное значение корня
$e_k = x_k - x^*$	ошибка приближенного значения корня
ε_p	требуемая точность вычисления приближенного значения корня
$f', f'', f^{(k)}$	производные функции $f(x)$

Решение уравнения $f(x)=0$ часто встречающаяся проблема. На первый взгляд она выглядит довольно простой, но точное её решение возможно, только если $f(x)$ есть полином степени $n \leq 4$. Под точным решением понимается некоторая процедура вычисления корня через параметры уравнения (например, для уравнения $ax^2 + bx + c = 0$).

Каждая нелинейная задача имеет свои особенности, поэтому численное решение уравнения $f(x)=0$ невозможно без предварительного анализа функции $f(x)$. Например, дополнительные трудности возникают когда график функции в точке корня касается оси x , как показано на рисунке 1.1. Такие корни трудно обнаружить, потому что график функции $f(x)$ не пересекает ось x . Такое поведение характерно для функций, которые имеют кратные корни. Корень уравнения x^* имеет кратность m если существует некоторая непрерывная функция $g(x)$ такая, что

- 1) $g(x^*) \neq 0$,
- 2) для каждого x , $f(x) = (x - x^*)^m g(x)$.

Если $m=1$, тогда x^* есть простой корень уравнения $f(x)=0$. Мы ограничимся рассмотрением случая, когда $f(x)$ вещественная однозначная функция вещественного аргумента и уравнение $f(x)=0$ имеет простой корень.

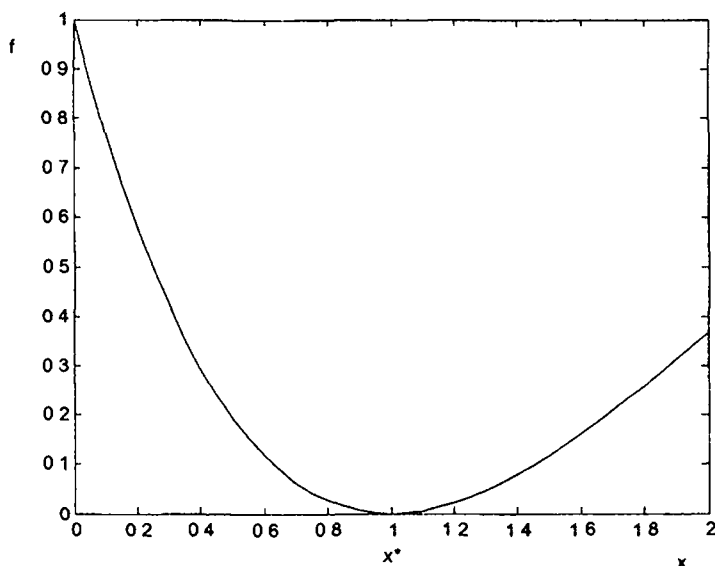


Рисунок 1.1. Ситуация когда $f(x^*) = f'(x^*) = 0$.

Основой для вычисления корня уравнения является итерационный процесс, или другими словами, процесс последовательных приближений. Итерационный процесс имеет четыре стадии.

- 1) найдем интервал, который содержит единственный корень (локализация корня)
- 2) выберем начальное приближение x_0
- 3) используя какую-либо итерационную процедуру k раз, получим набор приближений x_k , начиная с x_0 .
- 4) определим насколько близко каждое приближение x_k к точному значению корня. Если некоторое приближение находится в ε_p -окрестности x^* , тогда итерационный процесс завершен.

Последний пункт будет выполнен, только если

$$\lim_{k \rightarrow \infty} x_k = x^*,$$

то есть приближения x_k сходятся к x^* . Поэтому, основное внимание мы будем уделять условиям, которые обеспечивают сходимость итерационного процесса.

1.1. Метод деления отрезка пополам.

Рассмотрим простой, но надежный метод для вычисления корней. Пусть некоторый отрезок $[a, b] = I$ содержит единственный корень. Стратегия этого метода состоит в том, что мы делим интервал I пополам и затем оставляем только половину этого интервала содержащую корень. Очевидно, что половина, на которой функция $f(x)$ меняет знак, содержит корень. Это дает нам способ определить какую половину оставить и затем использовать для дальнейших вычислений. Метод деления отрезка пополам можно представить в виде блок-схемы показанной на рисунке 1.2.

Таким образом, на каждом шаге вычислительного процесса x^* лежит внутри интервала $[a_k, b_k]$, и как только длина этого интервала становится меньше чем ε_p , итерационный процесс завершается. В этом методе используется минимальная информация о функции $f(x)$ (знак $f(x)$). Поэтому требуется относительно большое число итераций, но сходимость этого метода гарантирована. Оценим количество итераций необходимых для вычисления корня. Легко видеть, что длина интервала $[a_k, b_k]$ уменьшается в два раза на каждом шаге итерационного процесса. Вычисления завершаются, когда

$$a_k - b_k = (a - b)/2^k \leq \varepsilon_p.$$

Для оценки числа итераций k выберем знак "=", тогда

$$k = \left\lceil \frac{\ln(b - a) - \ln \varepsilon_p}{\ln 2} \right\rceil + 1,$$

где $[y]$ означает целую часть y . Например, если $b - a = 1$ и $\varepsilon_p = 10^{-5}$ мы получим $k = 17$.

Пример 1.1 (метод деления отрезка пополам).

$$f(x) = \exp(-x) - \sin(x) = 0, \quad x^* \in [0; 1] = I, \quad \varepsilon_p = 10^{-5}.$$

Результаты расчетов представлены в Таблице 1.1.

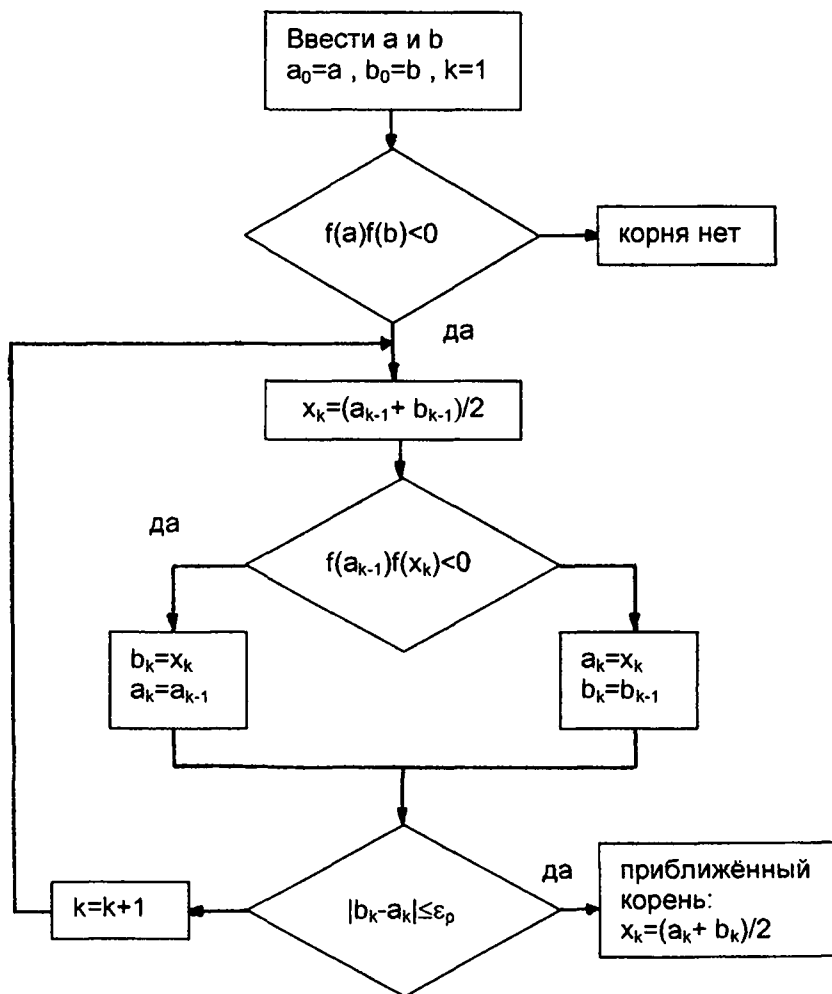


Рисунок 1.2. Блок-схема представляющая метод деления отрезка пополам

Таблица 1.1

k	x_k	$b_k - a_k$	$ f(x_k) $
1	0.5	5.00000e-01	1.27105e-01
2	0.75	2.50000e-01	2.09272e-01
3	0.625	1.25000e-01	4.98358e-02
4	0.5625	6.25000e-02	3.64801e-02
..			
17	0.58854	7.62939e-06	8.84776e-06

1.2 Вычисление корней с использованием итерационных функций.

В дальнейшем мы будем рассматривать другой подход для расчета корней. Уравнение $f(x)=0$ может быть представлено в следующей эквивалентной форме:

$$x = g(x), \quad (1.1)$$

где $g(x)$ - некоторая итерационная функция и $f(x^*)=0$ соответствует $x^*=g(x^*)$. Оказалось, что итерационные алгоритмы проще конструировать на основе эквивалентной формы (1.1). Итерационный процесс определяется следующим образом: зададим начальное значение x_0 , и будем вычислять последующие приближения по формуле

$$x_{k+1} = g(x_k), \quad k = 0, 1, \dots \quad (1.2)$$

На рисунке 1.3 показана геометрическая интерпретация сходящегося итерационного процесса (1.2). Следующая теорема объясняет условия, которые обеспечивают сходимость итераций (1.2).

Теорема 1.1.

Пусть $I=[a,b]$ ограниченный интервал и $g(x)$ функция, действующая из I в I . В дополнение, $g(x)$ удовлетворяет условию Липшица

$$|g(y) - g(z)| \leq L |y - z|, \quad 0 \leq L < 1$$

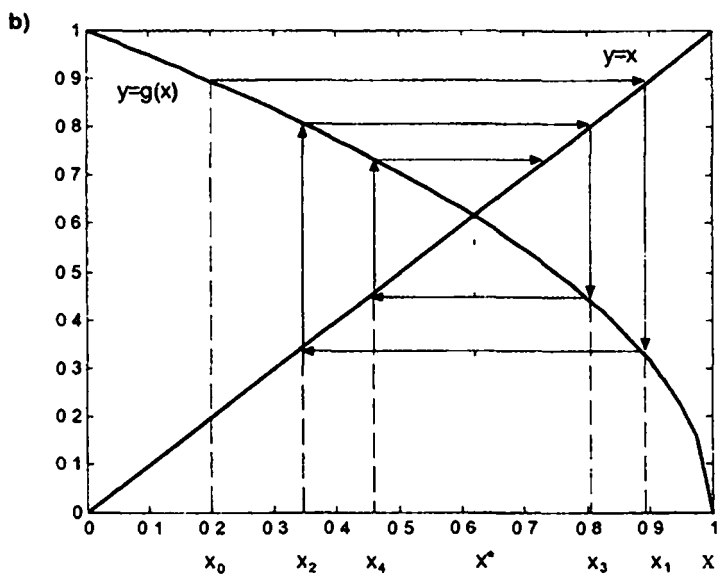
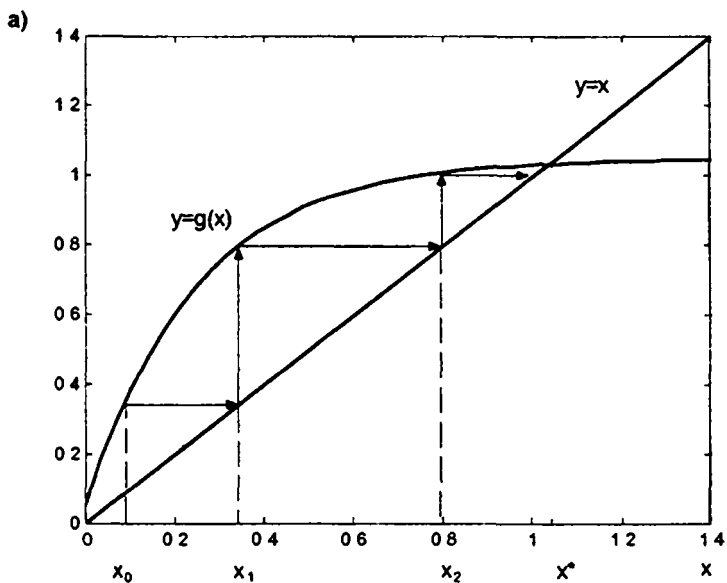


Рисунок 1.3 Последовательные приближения, генерируемые итерационным процессом (1.2): а) $g'(x) > 0$ и б) $g'(x) < 0$.

для произвольных точек y и z в I . Пусть $x_0 \in I$ и $x_{k+1} = g(x_k)$, тогда последовательность $\{x_k\}$ сходится к единственному решению $x^* \in I$ уравнения $x = g(x)$.

На основе этой теоремы можно получить оценку ошибки k -го приближения, которая зависит от двух последовательных приближений и константы Липшица L . Так как $|x_{k+1} - x_k| = |g(x_k) - g(x_{k-1})|$, мы можем заключить из условия Липшица, что для всех k , $|x_{k+1} - x_k| \leq L |x_k - x_{k-1}|$. Пусть n произвольное положительное целое число. Тогда

$$x_{k+n} - x_k = (x_{k+n} - x_{k+n-1}) + (x_{k+n-1} - x_{k+n-2}) + \dots + (x_{k+1} - x_k)$$

и

$$|x_{k+n} - x_k| \leq |x_{k+n} - x_{k+n-1}| + |x_{k+n-1} - x_{k+n-2}| + \dots + |x_{k+1} - x_k|$$

Следовательно, мы можем записать

$$|x_{k+n} - x_k| \leq L(1 + L + \dots + L^{n-1}) |x_k - x_{k-1}| = \frac{L(1 - L^n)}{1 - L} |x_k - x_{k-1}|$$

Пусть $n \rightarrow \infty$, тогда

$$|x^* - x_k| \leq \frac{L}{1 - L} |x_k - x_{k-1}| \quad (1.3)$$

Это дает нам оценку ошибки k -го приближения. Следует заметить, что если константа L близка к единице, то сходимость становится очень медленной. Поэтому основное наше внимание будет уделяться вопросу построения быстросходящихся итерационных процессов.

Имеется большое разнообразие методов для решения уравнения $f(x) = 0$. Мы рассмотрим лишь основные подходы, основанные на следующих трех типах итерационных процессов:

1) одноточечная итерация

$$x_{k+1} = g(x_k), \quad k = 0, 1, \dots \quad (1.4)$$

2) многоточечная итерация

$$x_{k+1} = g(x_k, \beta_1(x_k), \dots, \beta_n(x_k)), k = 0, 1, \dots, \quad (1.5)$$

где $\beta_1, \dots, \beta_n$ некоторые функции.

3) одноточечная итерация с памятью

$$x_{k+1} = g(x_k, x_{k-1}, \dots, x_{k-n}), k = n, n-1, \dots \quad (1.6)$$

Теперь следует разъяснить термин «быстрая сходимость» более подробно. Сделаем это на примере итерационного процесса (1.4). Сначала введем ошибку k -го приближения как $e_k = x_k - x^*$, тогда $x_k = e_k + x^*$ и $x_{k+1} = e_{k+1} + x^*$. После подстановки этих выражений в (1.4), мы можем разложить $g(x + e_k)$ по степеням e_k в окрестности точки x^* (предполагая, что все производные функции $g(x)$ до порядка p включительно, непрерывны):

$$\begin{aligned} x^* + e_{k+1} &= g(x^* + e_k) = g(x^*) + g'(x^*)e_k + \\ &+ \frac{1}{2}g''(x^*)e_k^2 + \dots + \frac{1}{p!}g^{(p)}(x^*)e_k^p, \\ y_k &\in [x_k, x^*] \end{aligned}$$

Учитывая, что $x^* = g(x^*)$ мы получаем

$$e_{k+1} = g'(x^*)e_k + \frac{1}{2}g''(x^*)e_k^2 + \dots + \frac{1}{p!}g^{(p)}(x^*)e_k^p \quad (1.7)$$

Если $g^{(n)}(x^*)=0$ для $n=1, \dots, p-1$ и $g^{(p)}(x^*) \neq 0$, тогда итерационная функция $g(x)$ имеет порядок p и как следует из (1.7)

$$e_{k+1} = C e_k^p, C = \text{const} \neq 0 \quad (1.8)$$

где p называется порядком последовательности $\{x_k\}$ и C называется константой асимптотической сходимости. Когда $p=1$, итерационная последовательность имеет линейную сходимость к x^* , а при $p>1$

сходимость сверхлинейная. Легко видеть, что если e_k достаточно мала, тогда при $p > 1$ итерационная последовательность может сходиться очень быстро.

Нам осталось обсудить один практически важный вопрос: как оценить ошибку k -го приближения к точному решению x^* . Это можно сделать на основе неравенства (1.3). Если $x_k \in [x - \varepsilon_p, x + \varepsilon_p]$, то выполняются следующие условия:

$$\begin{aligned} |x_k - x_{k-1}| &\leq \varepsilon_p \quad (\text{оценка абсолютной ошибки}) \\ \frac{|x_k - x_{k-1}|}{|x_k|} &\leq \varepsilon_p \quad (\text{оценка относительной ошибки}) \end{aligned} \quad (1.9)$$

В дополнение к условию (1.9), обычно требуют также выполнения условия

$$|f(x_k)| \leq \varepsilon_p.$$

Это условие следует из постановки проблемы. Таким образом, итерационный процесс завершается, когда одно из этих условий выполняется.

1.2.1. Одноточечный итерационный процесс.

Вначале мы рассмотрим случай когда $g'(x^*) \neq 0$. Как это следует из (1.7) $e_{k+1} = g'(x^*)e_k$ и при условии

$$|g'(x^*)| < 1 \quad (1.10)$$

будет линейная сходимость. Условие (1.10) невозможно использовать на практике, но мы можем потребовать выполнения следующего условия:

$$\max_{x \in I} |g'(x)| < 1, \quad x^* \in I, \quad (1.11)$$

Тогда теоретическое условие (1.10) будет также выполнено.

Простейший способ представить исходное уравнение в виде (1.1) следующий:

$$x = x + \alpha f(x) = g(x), \alpha = \text{const} \neq 0 \quad (1.12)$$

что дает итерационную схему

$$x_{k+1} = x_k + \alpha f(x_k) = g(x_k), k = 0, 1, \dots \quad (1.13)$$

Эта схема называется методом простой итерации. Для того, чтобы условие (1.11) выполнялось, параметр α должен подчиняться условиям:

$$0 < \alpha < \frac{2}{\max_{x \in I} |f'(x)|}, \text{ если } f'(x) < 0$$

$$-\frac{2}{\max_{x \in I} f'(x)} < \alpha < 0, \text{ если } f'(x) > 0$$

Для определённости можно выбрать

$$\alpha = \begin{cases} \frac{1}{\max_{x \in I} |f'(x)|}, & f'(x) < 0 \\ -\frac{1}{\max_{x \in I} f'(x)}, & f'(x) > 0 \end{cases}$$

Начальное приближение x_0 следует выбирать из интервала I .

Пример 1.2. (метод простой итерации).

$$f(x) = \exp(-x) - \sin(x) = 0,$$

$$x^* \in [0, \pi/2] = I.$$

Параметр α определяется как

$$\alpha = \frac{1}{\max_{x \in I} (\exp(-x) + \cos(x))} = 0.5.$$

Начальное приближение $x_0=0$ и требуемая точность $\epsilon_p=10^{-5}$. Результаты расчетов показаны в Таблице 1.2.

Таблица 1.2.

k	x_k	$ x_k - x_{k-1} $	$ f(x_k) $
1	0.5	5.00000e-01	1.27105e-01
2	0.563552	6.35525e-02	3.49906e-02
3	0.581047	1.74953e-02	1.04118e-02
4	0.586253	5.20590e-03	3.16351e-03
...			
9	0.588526	1.39279e-05	8.51199e-06

Результаты из примера 1.2 показывают, что сходимость итерационной последовательности в методе простой итерации не такая быстрая. Существует специальная процедура, которая позволяет ускорить сходимость, используя приближения полученные методом простой итерации. Выражение (1.7) для метода (1.13) может быть записано в виде

$$e_{k+1} = Re_k + O(e_k^2), \quad R = g'(x^*)$$

или

$$x_{k+1} - x^* = R(x_k - x^*) + O(e_k^2)$$

Рассмотрим две последовательные итерации

$$x_k - x^* = R(x_{k-1} - x^*) + O(e_{k-1}^2)$$

$$x_{k+1} - x^* = R(x_k - x^*) + O(e_k^2)$$

После исключения R , мы можем выразить x^* из этих двух уравнений:

$$x^* = \frac{x_k^2 - x_{k+1}x_{k-1}}{2x_k - x_{k+1} - x_{k-1}} + O(e_k^2) = z_{k+1} + O(e_k^2) \quad (1.14)$$

Если мы возьмём z_{k+1} в качестве приближения к x^* , тогда точность этого приближения будет $O(e_k^2)$, то есть, последовательность $\{z_k\}$ имеет сходимость второго порядка, в то время как последовательность $\{x_k\}$

имеет линейную сходимость. Преобразование $\{x_k\} \rightarrow \{z_k\}$ называется процессом Эйткена.

Пример 1.3. (процесс Эйткена).

Рассмотрим процесс Эйткена основанный на данных из примера 1.2. Результаты расчетов представлены в Таблице 1.3.

Таблица 1.3.

k	x_k	z_k	$ z_k - z_{k-1} $	$ f(z_k) $
1	0.50000			
2	0.56355			
3	0.58104	0.58769		1.16407e-03
4	0.58625	0.58845	7.65420e-04	1.02092e-04
5	0.58831	0.58852	6.67380e-05	9.50398e-06

Если сравнить эти результаты с результатами из примера 1.2, то заметно, что достигается некоторое ускорение сходимости

Как можно увидеть из предыдущего примера, сверхлинейная сходимость приводит к сокращению числа итераций. Однако, это достигается на основе преобразования итерационной последовательности с линейной сходимостью. Можно ли построить итерационный процесс, который непосредственно дает последовательность, сходящуюся к корню сверхлинейно? Имеется множество способов построения таких процессов, и далее мы рассмотрим один из них. Пусть производные $f(x)$ до порядка s включительно непрерывны на отрезке I . Тогда, применяя разложение по формуле Тейлора функции $f(x)$ в окрестности $x_k \in I$, получим

$$f(x) = P_s(x) + \frac{f^{(s)}(y_k(x))}{s!} (x - x_k)^s, \quad (1.15)$$

$$P_s(x) = \sum_{n=0}^{s-1} \frac{f^{(n)}(x_k)}{n!} (x - x_k)^n,$$

где y_k лежит в интервале, который зависит от x_k и x . Следующее приближение к x^* находится из уравнения (рисунок 1.4)

$$P_s(x_{k+1}) = 0. \quad (1.16)$$

Таким образом, выражая x_{k+1} из уравнения (1.16), мы получим отображение $x_k \rightarrow x_{k+1}$, что и определяет некоторую итерационную схему. Можно показать, что последовательность, генерируемая итерационной схемой (1.16) имеет порядок $p=s$.

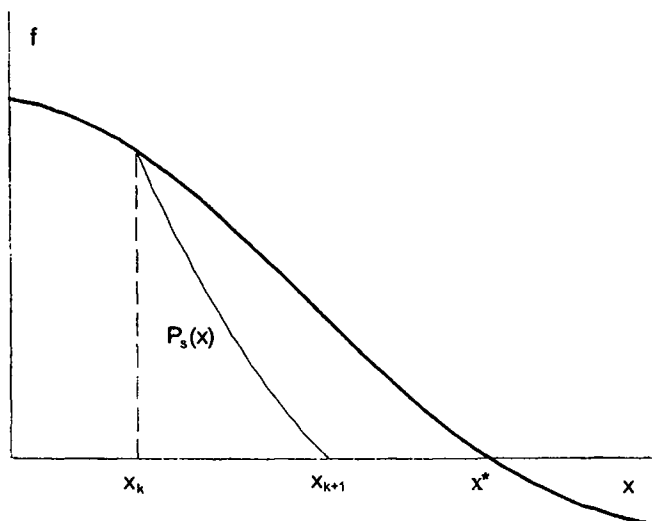


Рисунок 1.4. Графическое представление итерационной схемы основанной на разложении (1.15).

Теперь нам необходимо обсудить важный вопрос о том, как выбрать начальное приближение x_0 , которое дает сходящийся итерационный процесс. Пусть $x \in I$, производные $f^{(s)}(x)$ непрерывны на отрезке I и $f'(x)f^{(s)}(x) \neq 0$ для всех $x \in I$. Если условия

$$f(x_0)f^{(s)}(x_0) > 0, \text{ для четных } s \quad (1.17)$$

$$f'(x_0)f^{(s)}(x_0) < 0, \text{ для нечетных } s \quad (1.18)$$

выполняются и $\min(x^*, x_k) < x_{k+1} < \max(x^*, x_k)$, тогда итерационная последовательность $\{x_k\}$, генерируемая итерационной схемой (1.16) монотонно сходится к x^* .

Теперь рассмотрим конкретные методы, которые следуют из общего подхода (1.15) и (1.16). Когда $s=2$ уравнение (1.16) имеет следующий вид:

$$f(x_k) + f'(x_k)(x_{k+1} - x_k) = 0$$

Решая это уравнение относительно x_k , мы получим итерационную схему

$$x_{k+1} = x_k - \frac{f(x_k)}{f'(x_k)}, \quad k=0, 1, \dots \quad (1.19)$$

где

$$g(x) = x - \frac{f(x)}{f'(x)}$$

Это хорошо известный метод Ньютона и он имеет простую геометрическую интерпретацию. Функция

$$y(x) = f(x_k) + f'(x_k)(x - x_k)$$

представляет собой касательную к графику $f(x)$ в точке x_k . Тогда приближение x_{k+1} есть точка пересечения касательной с осью x . Для того, чтобы выбрать начальное приближение x_0 , необходимо использовать условие (1.17) при $s=2$.

Пример 1.4 (метод Ньютона).

$$f(x) = \exp(-x) - \sin(x) = 0, \quad x^* \in [0, \pi/2] = I.$$

$$f'(x) = -(\exp(-x) + \cos(x))$$

$$f''(x) = \exp(-x) + \sin(x)$$

Условие (1.17) для данной функции записывается, как

$$\exp(-2x_0) - \sin^2(x_0) > 0$$

и оно выполняется, например, если $x_0 \leq 0.5$. Выберем $x_0 = 0$ и $\varepsilon_p = 10^{-5}$. Результаты расчетов приведены в Таблице 1.4.

Таблица 1.4

k	x_k	$ x_k - x_{k-1} $	$ f(x_k) $
1	0.5	5.00000e-01	1.27105e-01
2	0.585643	8.56438e-02	4.01126e-03
3	0.588529	2.88558e-03	4.62670e-06

Легко видеть, что скорость сходимости значительно выше по сравнению с методом простой итерации.

В случае $s=3$ уравнение (1.16) принимает следующую форму:

$$f(x_k) + f'(x_k)(x_{k+1} - x_k) + 0.5f''(x_k)(x_{k+1} - x_k)^2 = 0 \quad (1.20)$$

Решая это уравнение относительно x_{k+1} , получим

$$x_{k+1} = x_k - \frac{f'(x_k)}{f''(x_k)} \pm \frac{|f'(x_k)|}{f''(x_k)} \left(1 - \frac{2f''(x_k)f(x_k)}{(f'(x_k))^2} \right)^{1/2} \quad (1.21)$$

Для того, чтобы условия Теоремы 1.1 выполнялись, необходимо взять знак “+” когда $f'(x) > 0$ и знак “-” когда $f'(x) < 0$. Также необходимо учесть, что сумма последних двух слагаемых в выражении (1.21) стремится к нулю вблизи корня. Это может привести к потере точности. Известно, что для квадратного уравнения $ax^2 + bx + c = 0$, справедливо следующее выражение:

$$\frac{1}{2a} \left(-b + \sqrt{b^2 - 4ac} \right) = -\frac{2c}{b + \sqrt{b^2 - 4ac}}$$

Применяя это преобразование к выражению (1.21), мы окончательно получим следующую итерационную схему:

$$x_{k+1} = x_k - \frac{f(x_k)}{f'(x_k) \left(1 + \sqrt{1 - \frac{2f''(x_k)f(x_k)}{(f'(x_k))^2}} \right)}, \quad k = 0, 1, \dots \quad (1.22)$$

Начальное условие x_0 следует выбирать из условия (1.18) при $s=3$.

Пример 1.5. (итерационная схема (1.22)).

$$f(x) = \exp(-x) - \sin(x) = 0, \quad x^* \in [0, \pi/2] = I.$$

$$f'(x) = -(\exp(-x) + \cos(x))$$

$$f''(x) = \exp(-x) + \sin(x)$$

$$f'''(x) = -\exp(-x) + \cos(x)$$

Условие (1.18) для данной функции записывается, как

$$\exp(-2x_0) - \cos^2(x_0) < 0$$

и оно выполняется, например, если $0 < x_0 < 1$. Выберем $x_0 = 0.001$ и $\varepsilon_p = 10^{-5}$. Результаты расчетов приведены в Таблице 1.5.

Таблица 1.5.

k	x_k	$ x_k - x_{k-1} $	$ f(x_k) $
1	0.585786	5.85686e-01	3.81299e-03
2	0.588532	2.74628e-03	2.57450e-09

Этот пример демонстрирует итерационный процесс с очень быстрой сходимостью.

Полином $P_s(x_{k+1})$ можно превратить в полином первой степени, и при этом порядок итерационной последовательности не изменится. Уравнение (1.20) можно модифицировать двумя способами:

- 1) заменить один из сомножителей $(x_{k+1} - x_k)$ в $(x_{k+1} - x_k)^2$ на $-f(x_k)/f'(x_k)$. После решения модифицированного уравнения относительно $(x_{k+1} - x_k)$, мы получим следующую итерационную схему:

$$x_{k+1} = x_k - \frac{f(x_k)f'(x_k)}{(f'(x_k))^2 - 0.5f''(x_k)f(x_k)}, \quad (1.23)$$

$$k = 0, 1, \dots$$

Это метод Хэлли(Halley).

2) заменить $(x_{k+1} - x_k)^2$ на $(f(x_k)/f'(x_k))^2$. Тогда после решения уравнения относительно $(x_{k+1} - x_k)$ мы получим

$$x_{k+1} = x_k - \frac{f(x_k)}{f'(x_k)} \left(1 - \frac{0.5f''(x_k)f(x_k)}{(f'(x_k))^2} \right), \quad (1.24)$$

$$k = 0, 1, \dots$$

С вычислительной точки зрения эти методы более привлекательны по сравнению с итерационной схемой (1.22) потому, что они также имеют кубическую сходимость, но исключают вычисление квадратного корня.

1.2.2 Многоточечный итерационный процесс.

Для того, чтобы построить итерационную схему с кубической сходимостью на основе одноточечной итерации, нам необходимо использовать значения второй производной функции $f(x)$. Иногда это может приводить к неоправданным вычислительным затратам. Существует другой способ достижения быстрой сходимости – это использовать многоточечные итерационные схемы (1.5). Одна из процедур построения таких схем основана на суперпозиции. Будем вычислять последующее приближение x_{k+1} в два этапа:

$$y_k = x_k - \frac{f(x_k)}{f'(x_k)} \quad (\text{метод Ньютона}),$$

$$x_{k+1} = y_k - \frac{f(y_k)}{f'(y_k)}$$

Комбинация этих двух формул приводит к следующей итерационной схеме:

$$x_{k+1} = x_k - \frac{\left(f(x_k) + f\left(x_k - \frac{f(x_k)}{f'(x_k)} \right) \right)}{f'(x_k)}, \quad (1.25)$$

$$k = 0, 1, \dots$$

$$\beta(x) = x - \frac{f(x)}{f'(x)}, \quad g(x) = \beta(x) - \frac{f(\beta(x))}{f'(x)}$$

Какую скорость сходимости дает этот метод? Для того, чтобы ответить на этот вопрос обратимся к выражению (1.7). Учитывая, что $\beta(x) = x$ и $\beta'(x) = 0$, можно вывести, что $g'(x) = g''(x) = 0$, но $g'''(x) \neq 0$. Это значит, что итерационная последовательность имеет порядок три, то есть, $e_{k+1} = C(e_k)^3$. В некоторых случаях многоточечная итерационная схема более эффективна по сравнению с одноточечной итерацией. Схема (1.25) использует только значения самой функции и первой производной и, в тоже время, обеспечивает кубическую сходимость. Способ построения итерационной схемы (1.25) позволяет нам применять такие же условия для выбора начального приближения, как и в методе Ньютона.

Пример 1.6(многоточечная итерационная схема (1.25)).

$$f(x) = \exp(-x) - \sin(x) = 0, \quad x^* \in [0, \pi] = I.$$

$$f'(x) = -(\exp(-x) + \cos(x))$$

Начальное приближение такое же, как в примере 1.4 ($x_0=0$) и $\varepsilon_p=10^{-5}$. Результаты расчетов приведены в Таблице 1.6.

Таблица 1.6

k	x_k	$ x_k - x_{k-1} $	$ f(x_k) $
1	0.563552	5.63552e-01	3.49906e-02
2	0.588528	2.49754e-02	6.52801e-06

Как можно видеть, мы имеем немного более медленную сходимость по сравнению с итерационной схемой (1.22) (пример 1.5). Тем не менее, двух итераций оказалось достаточно для того, чтобы достичь требуемой точности.

По аналогии с (1.25), можно также построить другую итерационную схему:

$$y_k = x_k - \frac{f(x_k)}{f'(x_k)},$$

$$x_{k+1} = y_k - \frac{f(y_k)}{f'(y_k)}$$

и после комбинирования этих двух выражений, мы получим

$$x_{k+1} = x_k - f(x_k) \left(\frac{1}{f'(x_k)} + \frac{1}{f' \left(x_k - \frac{f(x_k)}{f'(x_k)} \right)} \right),$$

$$k = 0, 1, \dots$$

Эта итерационная схема также имеет порядок сходимости $p=3$.

1.2.3 Одноточечный итерационный процесс с памятью

Методы, которые мы рассмотрели имеют высокую скорость сходимости, но их применение требует вычисления производных исходной функции, что может быть не всегда оправдано. Часто требуется значительно больше вычислительных затрат для того, чтобы вычислить $f'(x)$ и $f''(x)$ чем $f(x)$. В дополнение, во многих приложениях встречаются функции которые не имеют явных выражений для производных. Например, функция

$$f(x) = x + \int_0^1 \frac{\ln(y)}{1+x+y} dy$$

демонстрирует одну из возможных ситуаций. В этом параграфе мы рассмотрим некоторую модификацию одноточечной итерации, которая исключает вычисление производных функции. Первую производную функции $f(x)$ в точке x_k можно приближенно вычислить следующим образом:

$$f'(x_k) \approx \frac{f(x_k) - f(x_{k-1})}{x_k - x_{k-1}}$$

Подставляя это выражение в (1.19), мы получим

$$x_{k+1} = x_k - \frac{f(x_k)(x_k - x_{k-1})}{f(x_k) - f(x_{k-1})}, \quad (1.26)$$

$$k = 1, 2, \dots$$

Эта итерационная схема называется методом секущих. Сходимость этого метода может быть описана выражением (1.8) с некоторой константой C и $p \approx 1.62$, что показывает несколько более медленную сходимость по сравнению с методом Ньютона ($p=2$). Однако, метод секущих требует только одного вычисления $f(x_k)$ на итерацию (мы можем использовать значение $f(x_{k-1})$ из предыдущей итерации). Если вычисление $f(x)$ требует значительных вычислительных затрат, тогда использование метода Ньютона приводит к большому количеству вычислений на итерацию. В этом случае метод секущих может быть более эффективен: возможно, он потребует большего количества итераций, но количество вычислений на каждой итерации может быть значительно меньше. Метод секущих имеет свои особенности:

- 1) для того чтобы начать итерационный процесс нам необходимо задать два начальных приближения x_0 и x_1
- 2) числитель и знаменатель второго слагаемого в итерационной формуле (1.26) стремятся к нулю, когда x_k приближается к x^* . В этом случае ошибки округления могут значительно влиять на результат вычислений. Для того чтобы уменьшить это влияние, необходимо производить вычисления с двойной точностью.

Для выбора начальных значений x_0 и x_1 можно использовать такое же условие как для метода Ньютона, только применяя его к x_0 и x_1 :

$$f(x_m)f''(x_m) > 0, m=0, 1 \quad (1.27)$$

Пример 1.7 (метод секущих).

$$f(x) = \exp(-x) - \sin(x) = 0, x^* \in [0, \pi] = I.$$

$$f'(x) = -(\exp(-x) + \cos(x))$$

$$f''(x) = \exp(-x) + \sin(x)$$

Условие (1.27) для данной функции выполняется, например, при $x_0 = -0.01$ и $x_1 = 0$. Результаты расчетов для $\epsilon_p = 10^{-5}$ приведены в Таблице 1.7.

k	x_k	$ x_k - x_{k-1} $	$ f(x_k) $
2	0.49875	4.98753e-01	1.28956e-01
3	0.57259	7.38395e-02	2.22481e-02
4	0.58798	1.53952e-02	7.55890e-04
5	0.58853	5.41454e-04	4.79203e-06

Можно также получить формулу для приближенного вычисления $f''(x)$. Тогда, если подставить формулы для приближенного вычисления $f'(x)$ и $f''(x)$ в (1.22), (1.23) и (1.24), мы получим некоторые одноточечные итерационные процессы с памятью и n из (1.6) равняется двум. Эта процедура приводит к очень громоздким выражениям, в тоже время скорость сходимости этих методов лишь немного выше, чем в методе секущих (но ниже чем в методе Ньютона).

1.3 Заключительные замечания.

Выбор численного метода для решения уравнения $f(x)=0$ обусловлен в основном двумя факторами:

- 1) если необходимо решить одно уравнение, то можно использовать метод деления отрезка пополам или обратиться к процедуре из доступного пакета программ
- 2) если некоторая вычислительная процедура включает в себя многократное решение уравнения $f(x)=0$ то, возможно, потребуются использовать метод, который вычисляет приближенное значение корня наиболее эффективным образом.

Эффективность метода зависит от того, насколько просто значения производной функции $f'(x)$ могут быть вычислены. Если вычисление значений $f'(x)$ требует значительных затрат, тогда предпочтительнее использовать метод секущих. Если можно достаточно легко оценить значение $f'(x)$, тогда можно использовать метод простой итерации с процессом Эйткена. Например, рассмотрим следующее уравнение $f(x) =$

$J_0(x)-\alpha x$ ($J_0(x)$ и $J_1(x)$ функции Бесселя первого рода) и мы хотим найти $x \in [0,2]=I$. Первая производная исходной функции $f'(x) = -(J_1(x) + \alpha x)$. Из графика функции $J_1(x)$ видно, что

$$\max_{x \in I} J_1(x) \approx 0.6.$$

Таким образом, константа α в (1.13) определяется как $\alpha \approx 1/(a + 0.6)$.

Если затраты на вычисление значения $f'(x)$ сравнимы с затратами на вычисление значения $f(x)$, тогда метод Ньютона или многоточечная итерация будут наиболее эффективными.

Итерационные схемы третьего порядка (1.22), (1.23) and (1.24) требуют сравнительно большого количества операций на итерацию. Поэтому, эти методы могут найти применение только когда затраты на вычисление значений $f'(x_k)$ и $f''(x_k)$ относительно низки и необходимо определить корень с очень высокой точностью. Например, рассмотрим уравнение $f(x) = \exp(-x) - x = 0$. Вычисление производных $f'(x_k)$ и $f''(x_k)$ может быть организовано эффективным образом: значение $\exp(-x_k)$ сохраняется и затем используется для вычисления производных:

$$f(x_k) = \exp(-x_k) - x_k$$

$$f'(x_k) = -\exp(-x_k) - 1$$

$$f''(x_k) = \exp(-x_k)$$

ГЛАВА 2

Решение систем линейных уравнений

Список обозначений

$\mathbf{A}=\{a_{nm}\}$	матрица (заглавные буквы, жирный шрифт)
$\mathbf{x}$	вектор (строчные буквы, жирный шрифт)
N	размер матрицы или вектора
$\ \cdot\ $	норма вектора или матрицы
$\mathbf{A}^T, \mathbf{x}^T$	транспонированная матрица или вектор
$(\mathbf{x}, \mathbf{y})=\mathbf{x}^T \mathbf{y}$	скалярное произведение векторов $\mathbf{x}$ и $\mathbf{y}$
SPD	симметричная, положительно определенная (матрица) (Symmetric Positive Definite)
$\det(\mathbf{A})$	определитель матрицы
$\text{diag}(a_1, \dots, a_N)$	диагональная матрица с элементами $a_1, \dots, a_N$
$\mathbf{I}$	единичная матрица ($\mathbf{I}=\text{diag}(1, \dots, 1)$)
$\mathbf{A}^{-1}$	обратная матрица ($\mathbf{A}^{-1}\mathbf{A}=\mathbf{I}$)
ε_p	требуемая точность вычисления приближенного решения
e_p	ошибка возмущения
e_a	ошибка алгоритма
e_Γ	ошибка округления
$\mathbf{x}$	точное решение системы линейных уравнений ($\mathbf{Ax}=\mathbf{f}$)
$\mathbf{x}^{(k)}$	k-ое приближение к точному решению
$\mathbf{e}^{(k)}=\mathbf{x}^*-\mathbf{x}^{(k)}$	ошибка k-го приближения
$\mathbf{r}^{(k)}=\mathbf{Ax}^{(k)}-\mathbf{f}$	вектор невязки
$\mathbf{z}_n(\mathbf{B})$	n-ый собственный вектор матрицы $\mathbf{B}$
$\lambda_n(\mathbf{B})$	n-ое собственное значение матрицы $\mathbf{B}$
$s(\mathbf{B})=\max \lambda_n(\mathbf{B}) $	спектральный радиус матрицы $\mathbf{B}$

N_{ops}

общее количество операций (сложения и умножения/деления), необходимое для решения системы линейных уравнений

2.1 Необходимые сведения из линейной алгебры.

Вначале мы кратко рассмотрим некоторые основные определения из линейной алгебры, которые необходимы для понимания матричных вычислений. В дальнейшем мы будем предполагать, что матрицы и вектора вещественные. Тогда, вектора представляют собой элементы N -мерного Евклидова пространства R^N . В этом пространстве определены следующие операции:

сложение: $\mathbf{x} + \mathbf{y} = \mathbf{y} + \mathbf{x}$, $(\mathbf{x} + \mathbf{y}) + \mathbf{z} = \mathbf{x} + (\mathbf{y} + \mathbf{z})$

умножение на скаляр: $a(\mathbf{x} + \mathbf{y}) = a\mathbf{x} + a\mathbf{y}$, $\mathbf{x}, \mathbf{y}, \mathbf{z} \in R^N$

В дополнение, любой паре векторов $\mathbf{x}, \mathbf{y} \in R^N$ ставится в соответствие вещественное число $(\mathbf{x}, \mathbf{y})$, которое называется скалярным произведением. Для скалярного произведения постулируются следующие аксиомы:

$$(\mathbf{x}, \mathbf{y}) = (\mathbf{y}, \mathbf{x})$$

$$(\alpha \mathbf{x}, \mathbf{y}) = \alpha (\mathbf{x}, \mathbf{y}), \text{ здесь } \alpha \text{ вещественное число}$$

$$(\mathbf{x} + \mathbf{y}, \mathbf{z}) = (\mathbf{x}, \mathbf{z}) + (\mathbf{y}, \mathbf{z})$$

$$(\mathbf{x}, \mathbf{x}) > 0 \text{ при } \mathbf{x} \neq \mathbf{0}, \text{ здесь } \mathbf{0} \text{ обозначает нулевой вектор}$$

$$(\mathbf{0}, \mathbf{0}) = 0,$$

$$\mathbf{x}, \mathbf{y}, \mathbf{z} \in R^N$$

Векторы-столбцы

$$\mathbf{b}_1 = \begin{pmatrix} 1 \\ 0 \\ \cdot \\ \cdot \\ \cdot \\ 0 \end{pmatrix}, \mathbf{b}_2 = \begin{pmatrix} 0 \\ 1 \\ \cdot \\ \cdot \\ \cdot \\ 0 \end{pmatrix}, \dots, \mathbf{b}_N = \begin{pmatrix} 0 \\ \cdot \\ \cdot \\ \cdot \\ 0 \\ 1 \end{pmatrix}$$

образуют стандартный базис в $\mathbb{R}^N$. Все другие вектора могут быть разложены по базису $\mathbf{b}_n$ ($n=1, \dots, N$):

$$\mathbf{x} = x_1 \mathbf{b}_1 + \dots + x_N \mathbf{b}_N$$

Коэффициенты x_n представляют собой компоненты вектора $\mathbf{x}$ и это представление обычно записывается в компактной форме: $\mathbf{x} = (x_1, \dots, x_N)^T$. Надстрочный индекс T обозначает операцию транспонирования, которая превращает векторы-столбцы в векторы-строки и наоборот. Скалярное произведение двух векторов $\mathbf{x} = (x_1, \dots, x_N)^T$ и $\mathbf{y} = (y_1, \dots, y_N)^T$ вычисляется следующим образом

$$(\mathbf{x}, \mathbf{y}) = \mathbf{x}^T \mathbf{y} = \sum_{n=1}^N x_n y_n$$

Два вектора $\mathbf{x}$ и $\mathbf{y}$ ортогональны, если $(\mathbf{x}, \mathbf{y}) = 0$. Для того чтобы работать с каким-нибудь объектом, нам необходим способ его измерения. С этой целью вводится понятие нормы. Норма вектора $\mathbf{x}$ есть вещественное число $\|\mathbf{x}\|$ со следующими свойствами:

$$\|\mathbf{x}\| > 0 \text{ для } \mathbf{x} \neq \mathbf{0} \text{ и } \|\mathbf{0}\| = 0$$

$$\|\alpha \mathbf{x}\| = |\alpha| \cdot \|\mathbf{x}\|, \text{ здесь } \alpha \text{ некоторое вещественное число}$$

$$\|\mathbf{x} + \mathbf{y}\| \leq \|\mathbf{x}\| + \|\mathbf{y}\| \text{ для любых } \mathbf{x}, \mathbf{y}$$

Для вектора $\mathbf{x} = (x_1, \dots, x_N)^T$, вещественное число

$$\|\mathbf{x}\|_p = \left(\sum_{n=1}^N |x_n|^p \right)^{1/p}$$

есть норма при любом значении $p \geq 1$ (p -норма Гёльдера). На практике часто применяются следующие нормы:

$$\|\mathbf{x}\|_1 = \sum_{n=1}^N |x_n|,$$

$$\|\mathbf{x}\|_2 = \left(\sum_{n=1}^N |x_n|^2 \right)^{1/2} = \sqrt{(\mathbf{x}, \mathbf{x})},$$

$$\|\mathbf{x}\|_\infty = \max_n |x_n|.$$

Каждая квадратная матрица образует линейный оператор в векторном пространстве R^N . Если вектор y представляет собой линейную функцию вектора x , тогда

$$y = Ax, \quad A = \{a_{nm}\}$$

или

$$y_n = \sum_{m=1}^N a_{nm} x_m, \quad n=1, \dots, N$$

Правила операций над матрицами следуют из свойств линейных отображений. Так $C=A+B$ представляет собой сумму двух линейных функций заданных матрицами A и B , где $c_{nm}=a_{nm}+b_{nm}$. Произведение AB представляет собой результат применения отображения B , а затем отображения A . Если мы имеем $y=A(Bx)=Abx=Cx$, тогда компоненты матрицы C определяются по формуле

$$c_{nm} = \sum_{k=1}^N a_{nk} b_{km}, \quad n, m = 1, \dots, N$$

Транспонирование матрицы A осуществляется вращением A относительно её главной диагонали: $A^T = \{a_{nm}\}^T = \{a_{mn}\}$.

Диагональная матрица D имеет все нулевые компоненты, кроме компонент d_n на главной диагонали: $D=\text{diag}(d_1, \dots, d_N)$.

Теперь рассмотрим нормы матриц. Вещественное число $\|A\|$ есть норма матрицы, если выполняются следующие условия:

$$\begin{aligned} \|A\| &> 0 \text{ для } A \neq O, \text{ здесь } O \text{ обозначает нулевую матрицу и} \\ \|O\| &= 0 \\ \|\alpha A\| &= |\alpha| \cdot \|A\|, \text{ здесь } \alpha \text{ некоторое вещественное число} \\ \|A+B\| &\leq \|A\| + \|B\| \\ \|AB\| &\leq \|A\| \cdot \|B\| \end{aligned}$$

На практике часто применяются следующие нормы:

$$\begin{aligned} \|A\|_1 &= \max_m \sum_{n=1}^N |a_{nm}| \\ \|A\|_2 &= \left(\sum_{n=1}^N \sum_{m=1}^N a_{nm}^2 \right)^{1/2} \end{aligned}$$

$$\|A\|_{\infty} = \max_n \sum_{m=1}^N |a_{nm}|$$

В дальнейшем, символ $\|\cdot\|$ будет обозначать некоторую из рассмотренных выше норм. Матричная норма называется согласованной с векторной нормой, если $\|Ax\| \leq \|A\| \cdot \|x\|$. Например, $\|A\|_1$ согласована с $\|x\|_1$, $\|A\|_2$ с $\|x\|_2$, и $\|A\|_{\infty}$ с $\|x\|_{\infty}$.

Свойство определённости матрицы играет важную роль в матричных вычислениях. Так матрица A называется положительно (отрицательно) определённой, если $(x, Ax) > 0$ (< 0) для всех ненулевых векторов x . Если знак скалярного произведения (x, Ax) зависит от вектора x , тогда матрица A называется неопределённой.

2.2 Системы линейных уравнений.

Система линейных уравнений может быть представлена в виде:

$$Ax = f, \det(A) \neq 0, \quad (2.1)$$

или в развернутой форме

$$\begin{aligned} a_{11}x_1 + a_{12}x_2 + \dots + a_{1N}x_N &= f_1 \\ &\cdot \\ &\cdot \\ a_{N1}x_1 + a_{N2}x_2 + \dots + a_{NN}x_N &= f_N \end{aligned}$$

где $A = \{a_{nm}\}$ - квадратная матрица коэффициентов размером N на N , $x = (x_1, \dots, x_N)^T$ - вектор неизвестных размером N и $f = (f_1, \dots, f_N)^T$ - заданный вектор правой части размером N .

Некоторые физические задачи непосредственно приводят к системам линейных уравнений, например, статический анализ конструкций (мост, здание, несущая конструкция самолёта), проектирование электрических сетей. Многие физические процессы описываются дифференциальными уравнениями, которые часто не могут быть решены аналитически. В этом случае строится приближенный, дискретный аналог этих дифференциальных уравнений, и это часто приводит к системам линейных

уравнений (если дифференциальные уравнения линейные). Также системы линейных уравнений возникают как промежуточный этап в других вычислительных процессах. Например, одна из процедур решения систем нелинейных уравнений включает в себя решение набора систем линейных уравнений.

Классический метод решения систем линейных уравнений - метод Крамера, но он очень неэффективный и, следовательно, бесполезен с практической точки зрения. Для того чтобы решить этим методом систему с N неизвестными, необходимо вычислить $N+1$ определитель, а это потребует $\sim N^2 N!$ операций. Например, если $N=100$ (довольно небольшая система), тогда $N_{\text{ops}} \sim 10^{162}$. Обращение матрицы также неэффективный подход (если мы имеем A^{-1} , тогда вычисление решения x тривиально: $x=A^{-1}f$) потому, что обращение матрицы требует приблизительно в три раза больше вычислений и в два раза больше оперативной памяти по сравнению с другими методами решения систем линейных уравнений.

Для того чтобы выбрать тот или иной метод для решения системы (2.1) следует учитывать свойства матрицы A , такие как симметрия, определённость, ленточная структура и разреженность. Это позволяет решить систему линейных уравнений более эффективно. Выбор метода также зависит от типа решаемой проблемы, а именно:

тип 1) решить систему линейных уравнений $Ax=f$ один раз

тип 2) решить набор систем линейных уравнений $Ax_k = f_k$, $k = 1, \dots, M$, где матрица A остается неизменной для всех k .

2.3 Типы матриц, часто встречающиеся при решении задач.

1) разреженные матрицы.

Матрицы, большинство элементов которых нули, называются разреженными. Одно из определений разреженной матрицы следующее: матрица A размером N на N считается разреженной, если число её ненулевых элементов $\sim N^{1+\gamma}$ ($\gamma \leq 0.5$). Например, при $N=10^3$ и $\gamma=0.5$, число ненулевых элементов 31622 (общее число элементов 10^6).

2) ленточные матрицы.

Многие задачи приводят к матрицам, которые не только разрежены, но имеют ленточную структуру ненулевых элементов. Матрица $A=\{a_{nm}\}$

называется ленточной, если $a_{nm}=0$ при $|n - m|>k$ и $l=2k+1$ есть ширина ленты:

$$A = \begin{pmatrix} * & * & * & & & \\ * & . & . & * & & 0 \\ * & . & . & . & * & \\ * & \leftarrow & | & \rightarrow & * & \\ & * & . & . & . & * \\ 0 & & * & . & . & * \\ & & & * & * & * \end{pmatrix}$$

3) симметричные положительно определённые матрицы.

Многие физические задачи приводят к симметричным положительно определённым (SPD) матрицам. Такие матрицы имеют следующие свойства:

- 1) $A=A^T$ (симметрия)
- 2) $(x, Ax) > 0$ для всех ненулевых векторов x (положительная определённость).

Непосредственно использовать определение 2) для выяснения определённости матрицы невозможно. Однако существуют другие критерии, которые могут быть использованы на практике. Во-первых, все собственные значения положительно определенной матрицы положительны. Это может быть легко проверено с помощью теоремы Гершгорина (смотри Главу 3). Во-вторых, положительно определенная матрица A размером N на N имеет следующие свойства:

- а) $a_{nn} > 0$ для всех n
- б) $a_{nn}a_{mm} > (a_{nm})^2$ для $n \neq m$
- в) наибольший по модулю элемент должен лежать на главной диагонали

4) треугольные матрицы.

На практике часто возникают два типа треугольных матриц:

$$L = \begin{pmatrix} * & & & & \\ * & * & & & 0 \\ * & . & * & & \\ * & . & . & * & \\ * & * & * & * & * \end{pmatrix}$$

нижняя треугольная ($a_{nm} = 0$ при $m > n$),

$$U = \begin{pmatrix} * & * & * & * & * \\ & * & . & . & * \\ & & * & . & * \\ & & & * & * \\ 0 & & & & * \\ & & & & * \end{pmatrix}$$

верхняя треугольная ($a_{nm} = 0$ при $m < n$).

Система линейных уравнений $Lx=f$ может быть решена прямой подстановкой:

$$\begin{aligned} x_1 &= f_1 / a_{11} \\ x_n &= \frac{1}{a_{nn}} \left(f_n - \sum_{m=1}^{n-1} a_{nm} x_m \right), \quad n = 2, \dots, N \end{aligned} \quad (2.2)$$

Аналогично, система линейных уравнений $Ux=f$ решается с помощью обратной подстановки:

$$\begin{aligned} x_N &= f_N / a_{NN} \\ x_n &= \frac{1}{a_{nn}} \left(f_n - \sum_{m=n+1}^N a_{nm} x_m \right), \quad n = N-1, \dots, 1 \end{aligned} \quad (2.3)$$

5) ортогональные матрицы

Если матрица A ортогональная, тогда $A^T A = I$ или $A^T = A^{-1}$, и система линейных уравнений (2.1) решается очень легко: $x = A^T f$.

6) разложимые матрицы

Квадратная матрица A называется разложимой, если строки и столбцы этой матрицы можно переставить таким образом, что она представляется в следующем виде:

$$A = \begin{pmatrix} B & C \\ O & D \end{pmatrix},$$

где B – квадратная матрица размером M на M , D – квадратная матрица размером L на L , C – матрица размером M на L and O – нулевая матрица

размером L на M ($M + L = N$). Это свойство позволяет разбить исходную систему на две системы меньшего размера, что значительно уменьшает объем вычислений. Вначале разделим каждый из векторов $\mathbf{x}$ и $\mathbf{f}$ на два вектора:

$$\mathbf{x}_1 = (x_1, \dots, x_M)^T, \quad \mathbf{x}_2 = (x_{M+1}, \dots, x_N)^T$$

$$\mathbf{f}_1 = (f_1, \dots, f_M)^T, \quad \mathbf{f}_2 = (f_{M+1}, \dots, f_N)^T$$

Тогда система $\mathbf{Ax} = \mathbf{f}$ решается в два этапа:

а) $\mathbf{D}\mathbf{x}_2 = \mathbf{f}_2$

б) $\mathbf{B}\mathbf{x}_1 = \mathbf{f}_1 - \mathbf{C}\mathbf{x}_2$

7) матрицы с диагональным преобладанием

Матрица $\mathbf{A}$ называется матрицей со строгим диагональным преобладанием если

$$|a_{nn}| > \sum_{\substack{m=1 \\ m \neq n}}^N |a_{nm}| = \text{sum}_n, \quad n = 1, \dots, N.$$

Матрица $\mathbf{A}$ называется матрицей со слабым диагональным преобладанием если

$$|a_{nn}| \geq \text{sum}_n, \quad n = 1, \dots, N,$$

и, по крайней мере, одно из неравенств выполняется как строгое неравенство.

2.4 Источники ошибок.

Когда мы решаем задачу (2.1) используя какой-нибудь метод, мы не получим точного ответа, так как в процессе решения возникают различные ошибки. Предположим, что задача (2.1) поставлена точно, то есть, мы пренебрегаем неизбежными ошибками, связанными с формулированием задачи как модели некоторого явления (ошибки измерений, недостаточно точные основные предположения и т. д.).

Вектор $\mathbf{f}$ в (2.1) обычно не задается точно, а вычисляется через параметры исходной проблемы. Эти вычисления выполняются с

определённой ошибкой, и вместо вектора $\mathbf{f}$ мы получаем некоторый “возмущённый” вектор $\mathbf{f} + \delta\mathbf{f}$. Тогда вместо задачи (2.1), мы получим “возмущённую” задачу

$$\mathbf{A}(\mathbf{x} + \delta\mathbf{x}) = \mathbf{A}\mathbf{y} = \mathbf{f} + \delta\mathbf{f} \quad (2.4)$$

Предположим, что эта система также может быть решена и $\mathbf{y}$ есть решение этой системы. Тогда

$$e_p = \|\mathbf{y} - \mathbf{x}^*\|$$

называется ошибкой возмущения.

Теперь нам необходимо решить возмущенную систему (2.4). Предположим, что мы имеем некоторый алгоритм, который дает после конечного числа операций или вектор $\mathbf{y}$ (если операции выполняются точно) или некоторый вектор $\mathbf{z}$ такой, что ошибка

$$e_a = \|\mathbf{z} - \mathbf{y}\|$$

может быть сделана произвольно малой. Величина e_a называется ошибкой алгоритма.

Любой алгоритм включает в себя операции с плавающей точкой, которые выполняются неточно. Ошибки, возникающие при таких операциях, приводят к тому, что вместо вектора $\mathbf{z}$ мы получаем некоторый другой вектор $\mathbf{w}$ и величина

$$e_r = \|\mathbf{w} - \mathbf{z}\|$$

называется ошибкой округления.

Окончательно, вместо точного решения $\mathbf{x}^*$ мы вычислим некоторый вектор $\mathbf{w}$, который является приближенным решением задачи (2.1) и ошибка этого решения может быть оценена как

$$e = \|\mathbf{w} - \mathbf{x}^*\| \leq e_p + e_a + e_r$$

2.5 Число обусловленности.

Как мы обсуждали раньше, мы имеем исходную задачу $\mathbf{A}\mathbf{x} = \mathbf{f}$ и возмущенную задачу $\mathbf{A}(\mathbf{x} + \delta\mathbf{x}) = \mathbf{f} + \delta\mathbf{f}$, где $\delta\mathbf{f}$ представляет теперь суммарное

возмущение вектора правой части и матрицы $\mathbf{A}$. Наша задача состоит в том, чтобы выяснить, как возмущение $\delta \mathbf{f}$ передается в возмущение решения $\delta \mathbf{x}$, то есть, оценить величину

$$\frac{\|\delta \mathbf{x}\| / \|\mathbf{x}\|}{\|\delta \mathbf{f}\| / \|\mathbf{f}\|} \quad (2.5)$$

Перепишем это выражение в виде

$$\frac{\|\delta \mathbf{x}\| / \|\mathbf{x}\|}{\|\delta \mathbf{f}\| / \|\mathbf{f}\|} = \frac{\|\delta \mathbf{x}\| \cdot \|\mathbf{f}\|}{\|\delta \mathbf{f}\| \cdot \|\mathbf{x}\|}$$

Из выражений (2.1) и (2.4) следует, что $\|\mathbf{f}\| = \|\mathbf{A}\mathbf{x}\| \leq \|\mathbf{A}\| \cdot \|\mathbf{x}\|$ и $\|\delta \mathbf{x}\| = \|\mathbf{A}^{-1} \delta \mathbf{f}\| \leq \|\mathbf{A}^{-1}\| \cdot \|\delta \mathbf{f}\|$. Если подставить эти оценки в предыдущее выражение, мы получим

$$\frac{\|\delta \mathbf{x}\| / \|\mathbf{x}\|}{\|\delta \mathbf{f}\| / \|\mathbf{f}\|} \leq \|\mathbf{A}\| \cdot \|\mathbf{A}^{-1}\|$$

или

$$\frac{\|\delta \mathbf{x}\|}{\|\mathbf{x}\|} \leq \|\mathbf{A}\| \cdot \|\mathbf{A}^{-1}\| \frac{\|\delta \mathbf{f}\|}{\|\mathbf{f}\|}.$$

Величина

$$\text{cond}(\mathbf{A}) = \|\mathbf{A}\| \cdot \|\mathbf{A}^{-1}\|$$

называется стандартным числом обусловленности. Это число оценивает максимальный коэффициент усиления возмущения $\delta \mathbf{f}$. Если $\text{cond}(\mathbf{A})$ относительно мало, тогда система (2.1) хорошо обусловлена и возмущение решения $\delta \mathbf{x}$ также относительно мало. Например, если $\|\delta \mathbf{f}\| / \|\mathbf{f}\| \sim 10^{-14}$ и нас удовлетворяет $\|\delta \mathbf{x}\| / \|\mathbf{x}\| \sim 10^{-6}$, тогда $\text{cond}(\mathbf{A}) \sim 10^8$ вполне приемлемо. Если $\text{cond}(\mathbf{A})$ относительно большое, тогда мы сталкиваемся с плохо обусловленной системой и требуются специальные методы, для того чтобы получить приемлемый результат.

Пример 2.1 (плохо обусловленная система)

Рассмотрим следующую систему линейных уравнений:

$$A = \begin{pmatrix} 1 & -1 & . & . & . & -1 \\ & 1 & -1 & . & . & -1 \\ & & . & & . & . \\ & & & . & . & . \\ & 0 & & & 1 & -1 \\ & & & & & 1 \end{pmatrix}, f = \begin{pmatrix} -1 \\ . \\ . \\ . \\ -1 \\ 1 \end{pmatrix},$$

$$\det(A) = 1$$

Точное решение этой системы $x=(0, \dots, 0, 1)^T$. Предположим, что только последняя компонента вектора f возмущена: $\delta f=(0, \dots, 0, \delta)^T$. Матрица A верхняя треугольная, поэтому система (2.4) может быть легко решена относительно вектора δx . Учитывая что $Ax=f$, система (2.4) преобразуется в $A\delta x=\delta f$. Распишем эту систему более подробно ($\delta x=(p_1, \dots, p_N)^T$):

$$p_1 - p_2 - \dots - p_N = 0$$

$$p_2 - \dots - p_N = 0$$

$$\dots\dots\dots$$

$$p_{N-1} - p_N = 0$$

$$p_N = \delta$$

Данная система просто решается обратной подстановкой, которая дает следующее решение:

$$p_N = \delta$$

$$p_{N-1} = \delta$$

$$p_{N-2} = 2\delta$$

$$\dots\dots\dots$$

$$p_1 = 2^{N-2} \delta$$

Теперь мы имеем все вектора, чтобы вычислить коэффициент усиления определенный в (2.5). Выберем векторную норму $\|\cdot\|_\infty$, тогда

$$\|\delta x\|_\infty = 2^{N-2} |\delta|,$$

$$\|x\|_\infty = 1,$$

$$\|\delta f\|_{\infty} = \|\delta\|,$$

$$\|f\|_{\infty} = 1$$

и

$$\text{cond}(\mathbf{A}) \geq 2^{N-2}$$

Например, если $N=102$ и $\delta=10^{-14}$, тогда $\text{cond}(\mathbf{A}) \sim 10^{30}$ и $\|\delta x\|_{\infty} \sim 10^{16} \gg \|x\|_{\infty}$. Система (2.6) является примером очень плохо обусловленной системы.

Непосредственное вычисление числа обусловленности весьма затруднительно, так как требует вычисления обратной матрицы. Поэтому, обычно прибегают к оценке этого числа.

В некоторых случаях оценка для числа обусловленности может быть получена довольно просто. Пусть матрица $\mathbf{A}$ может быть представлена в следующем виде

$$\mathbf{A} = \mathbf{I} + \mathbf{H}, \quad (2.7)$$

где $\|\mathbf{H}\|_{\infty} < 1$. Тогда

$$(\mathbf{I} + \mathbf{H})^{-1} = \mathbf{I} + \sum_{k=1}^{\infty} (-\mathbf{H})^k$$

и

$$\|(\mathbf{I} + \mathbf{H})^{-1}\|_{\infty} \leq \frac{1}{1 - \|\mathbf{H}\|_{\infty}}$$

Следовательно

$$\text{cond}(\mathbf{A}) \leq \frac{\|\mathbf{I} + \mathbf{H}\|_{\infty}}{1 - \|\mathbf{H}\|_{\infty}} \leq \frac{1 + \|\mathbf{H}\|_{\infty}}{1 - \|\mathbf{H}\|_{\infty}}.$$

Матрицы со строгим диагональным преобладанием могут быть представлены в виде (2.7).

2.6 Прямые методы.

В этой части мы рассмотрим прямые методы. Отличительная особенность этих методов состоит в том, что, пренебрегая ошибками

округлений, они дают точное решение после конечного числа операций. Далее мы обсудим лишь основные принципы прямых методов.

2.6.1 Основные принципы прямых методов.

На первом этапе решения систем линейных уравнений матрица **A** представляется в виде произведения двух треугольных матриц. Эта операция называется разложением матрицы, и форма разложения зависит от свойств матрицы **A**. Для матриц общего вида используется LU-разложение

$$\mathbf{A} = \mathbf{LU},$$

где **L** - нижняя треугольная матрица с единицами на диагонали, и **U** - верхняя треугольная матрица. Для SPD матриц используется разложение Холецкого

$$\mathbf{A} = \mathbf{LL}^T,$$

где **L** - нижняя треугольная матрица. После разложения матрицы **A**, исходная система $\mathbf{Ax}=\mathbf{f}$ может быть представлена в виде двух систем уравнений:

$$\mathbf{Ly} = \mathbf{f}, \mathbf{Ux} = \mathbf{y}$$

или

$$\mathbf{Ly} = \mathbf{f}, \mathbf{L}^T \mathbf{x} = \mathbf{y}$$

(2.8)

Применяя прямую, а потом обратную подстановки, можно вычислить решение исходной системы (смотри (2.2) и (2.3)). При таком подходе, суммарная ошибка полученного решения состоит из ошибки возмущения в множителях **L** и **U**, и ошибки округлений при решении систем (2.8). Вычислительные затраты для метода LU-разложения составляют $N_{\text{ops}} \sim N^3$ (для матриц общего вида), а для разложения Холецкого требуется в два раза меньше вычислений и оперативной памяти чем для метода LU-разложения.

Существует ещё одно полезное разложение: найти ортогональную матрицу **Q** и верхнюю треугольную матрицу **R**, так что $\mathbf{QAx}=\mathbf{Rx}=\mathbf{Qf}$. Тогда решение **x** может быть получено обратной подстановкой при

векторе правой части равно Qf . Ортогональное разложение привлекательно тем, что умножение A или f на Q не увеличивает ошибки округления или возмущения связанные с матрицей A или вектором f . Для того чтобы убедиться в этом, заметим, что

$$(Qx, Qx) = (Qx)^T (Qx) = x^T Q^T Qx = x^T x$$

так как $Q^T Q = I$. Преимущество такого подхода состоит в том, что он дает вычислительно устойчивый метод. Однако, этот метод требует в два раза больше вычислений по сравнению с методом LU-разложения. Поэтому ортогональные разложения применяются для решения систем линейных уравнений только в некоторых специальных случаях. Ортогональные разложения также находят применение для вычисления собственных значений.

Далее мы рассмотрим более подробно простой и эффективный метод для решения систем линейных уравнений называемый методом прогонки. Этот метод основан на специальном разложении, которое применимо к системам уравнений с ленточной матрицей. Мы ограничимся случаем, когда матрица системы (2.1) трехдиагональная. Общая структура метода прогонки может быть представлена в следующем виде:

$$\begin{pmatrix} * & * & & 0 \\ * & * & * & \\ & \cdot & \cdot & \cdot \\ & & * & * & * \\ 0 & & & * & * \end{pmatrix} \begin{pmatrix} x_1 \\ \cdot \\ \cdot \\ \cdot \\ x_n \end{pmatrix} = \begin{pmatrix} * \\ \cdot \\ \cdot \\ \cdot \\ * \end{pmatrix} \quad \text{прямой ход} \rightarrow$$

$$\begin{pmatrix} 1 & + & & 0 \\ & \cdot & \cdot & \\ & & \cdot & \cdot \\ & & & \cdot & + \\ 0 & & & 1 \end{pmatrix} \begin{pmatrix} x_1 \\ \cdot \\ \cdot \\ \cdot \\ x_n \end{pmatrix} = \begin{pmatrix} + \\ \cdot \\ \cdot \\ \cdot \\ + \end{pmatrix} \quad \text{обратный ход} \rightarrow \begin{pmatrix} x_1 \\ \cdot \\ \cdot \\ \cdot \\ x_n \end{pmatrix}$$

Для удобства, запишем систему уравнений с трехдиагональной матрицей в виде:

$$\begin{aligned}
 b_1x_1 + c_1x_2 &= f_1 \\
 a_nx_{n-1} + b_nx_n + c_nx_{n+1} &= f_n, \quad n = 2, \dots, N-1 \\
 a_Nx_{N-1} + b_Nx_N &= f_N
 \end{aligned}
 \tag{2.9}$$

Предположим, что мы можем записать следующее рекуррентное соотношение для компонент вектора решения:

$$x_n = \alpha_n x_{n+1} + \beta_n, \tag{2.10}$$

по крайней мере, это возможно для первого уравнения. Пусть коэффициенты α_{n-1} и β_{n-1} известны, тогда мы можем записать

$$\begin{aligned}
 x_{n-1} &= \alpha_{n-1}x_n + \beta_{n-1} \\
 a_nx_{n-1} + b_nx_n + c_nx_{n+1} &= f_n
 \end{aligned}$$

После исключения x_{n-1} из этих двух соотношений, мы получим

$$x_n = -\frac{c_n}{a_n\alpha_{n-1} + b_n} x_{n+1} + \frac{f_n - a_n\beta_{n-1}}{a_n\alpha_{n-1} + b_n}$$

Сравнивая это соотношение с (2.10), мы видим, что коэффициенты α_n и β_n могут быть вычислены по формулам

$$\begin{aligned}
 \alpha_n &= -\frac{c_n}{a_n\alpha_{n-1} + b_n}, \\
 \beta_n &= \frac{f_n - a_n\beta_{n-1}}{a_n\alpha_{n-1} + b_n}, \quad n = 2, \dots, N \\
 \alpha_1 &= -\frac{c_1}{b_1}, \quad \beta_1 = \frac{f_1}{b_1},
 \end{aligned}$$

После того, как параметры α_n и β_n вычислены, решение системы (2.9) может быть легко получено. В начале, запишем соотношение (2.10) для $n=N-1$ и последнее уравнение (2.9):

$$\begin{aligned}
 x_{N-1} &= \alpha_{N-1}x_N + \beta_{N-1} \\
 a_Nx_{N-1} + b_Nx_N &= f_N
 \end{aligned}$$

Исключение x_{N-1} дает

$$x_N = \frac{f_N - a_N \beta_{N-1}}{a_N \alpha_{N-1} + b_N}.$$

Остальные компоненты x_n вычисляются с использованием формулы (2.10) для $n=N-1, \dots, 1$. Метод прогонки в полной мере учитывает ленточную структуру матрицы, поэтому этот метод требует всего лишь $N_{\text{ops}}=8N-6$.

Следующая теорема описывает условия, при которых метод прогонки устойчив.

Теорема 2.1

Предположим, что коэффициенты системы (2.9) удовлетворяют условиям

$$\begin{aligned} b_1 &\neq 0, \quad b_N \neq 0, \\ a_n &\neq 0, \quad c_n \neq 0 \quad (n = 2, \dots, N-1), \\ |b_n| &\geq |a_n| + |c_n| \quad n = 2, \dots, N-1, \\ |b_1| &\geq |c_1|, \quad |b_N| \geq |a_N|, \end{aligned} \quad (2.11)$$

где по крайней мере одно из неравенств из последней строки есть строгое неравенство. Тогда выполняются следующие неравенства:

$$|\alpha_n| \leq 1 \quad (n = 2, \dots, N)$$

и

$$|a_n \alpha_{n-1} + b_n| \geq \min_n |c_n|$$

Таким образом, условия (2.11) достаточны для устойчивости метода прогонки.

2.6.2 Оценка ошибки приближенного решения.

Непосредственно вычислить вектор ошибки $\mathbf{e} = \mathbf{x} - \mathbf{x}^*$ мы не можем, так как точное решение нам не известно. Мы можем, по крайней мере, с машиной точностью, вычислить вектор невязки $\mathbf{r} = \mathbf{Ax} - \mathbf{f}$, который обращается в нулевой вектор когда $\mathbf{e} = \mathbf{0}$. Однако, малая величина вектора невязки не гарантирует, что вектор ошибки также мал. Проанализируем взаимосвязь между $\mathbf{e}$ и $\mathbf{r}$ более подробно. В начале заметим, что $\mathbf{r} = \mathbf{Ax} - \mathbf{Ax}^* = \mathbf{A}(\mathbf{x} - \mathbf{x}^*) = \mathbf{Ae}$, то есть $\mathbf{e} = \mathbf{A}^{-1}\mathbf{r}$. Тогда отношение $\|\mathbf{e}\|/\|\mathbf{f}\|$ примет вид

$$\frac{\|e\|}{\|f\|} = \frac{\|e\|}{\|Ax^*\|} = \frac{\|A^{-1}r\|}{\|Ax^*\|} \leq \frac{\|A^{-1}\| \cdot \|r\|}{\|Ax^*\|}$$

Учитывая, что $\|Ax^*\| \leq \|A\| \cdot \|x^*\|$, мы можем заменить знаменатель в левой части неравенства и, умножая на $\|A\|$, получим оценку

$$\frac{\|e\|}{\|x^*\|} \leq \|A\| \cdot \|A^{-1}\| \frac{\|r\|}{\|f\|} = \text{cond}(A) \frac{\|r\|}{\|f\|}.$$

Это неравенство показывает, что относительная ошибка мала, когда относительная величина вектора невязки мала и число обусловленности не слишком велико.

2.6.3 Заключительные замечания.

Начиная с середины 50-х годов, был достигнут значительный прогресс в области вычислительной линейной алгебры. Это подтверждается наличием эффективного и надёжного программного обеспечения реализующего различные прямые методы для решения систем линейных уравнений. Это обеспечение доступно или в виде библиотеки подпрограмм (IMSL®, NAG®, LAPACK), или как встроенные функции системы программирования (например, MATLAB®). Эти программы реализуют различные формы разложения, которые мы обсуждали, для различных типов матриц, таких как общие, ленточные, симметричные, разреженные и SPD матрицы. На практике, накопление ошибок округления может приводить к значительному искажению вычисленного решения. Поэтому все алгоритмы в пакетах программ включают в себя специальные меры (такие как выбор ведущего элемента и балансировка матриц) для того, чтобы уменьшить влияние этого эффекта. В стандартном программном обеспечении также обеспечивается оценка числа обусловленности, что даёт возможность оценить ошибку вычисленного решения. В дополнение, библиотеки подпрограмм включают в себя пакет BLAS (Basic Linear Algebra Subprograms, основные подпрограммы линейной алгебры). Этот пакет реализует различные операции над векторами и матрицами, и он широко используется при создании прикладного программного обеспечения для научных и инженерных вычислений.

2.7 Итерационные методы.

Итерационные методы для задачи $Ax=f$ требуют бесконечного числа операций для получения точного решения (если предположить, что вычисления производятся точно). Конечно, на практике мы производим конечное число шагов и в результате получаем некоторое приближенное решение. Итерационные методы легко конструируются и могут быть очень эффективны. Однако, они требуют дополнительного анализа свойств матрицы для того, чтобы выбрать подходящий метод.

2.7.1 Основные принципы итерационных методов.

Итерационные методы обычно основываются на следующей эквивалентной форме системы (2.1):

$$x = Bx + g, \quad (2.12)$$

где B называется итерационной матрицей. Эта форма системы линейных уравнений образуется по аналогии с (1.1), мы просто применили тот же подход к системе (2.1). Компоненты итерационной матрицы выражаются через компоненты матрицы A . Вектор g выражается через вектор f и компоненты матрицы A . Матрица B в явном виде обычно не вычисляется. но мы будем подразумевать, что, в принципе, это может быть сделано. Итерационный процесс формулируется на основе эквивалентной формы (2.12) в следующем виде:

- 1) зададим начальное приближение (вектор) $x^{(0)}$
- 2) для каждого значения k будем вычислять последующие приближения как

$$x^{(k+1)} = Bx^{(k)} + g \quad (2.13)$$

или

$$x^{(k+1)} = B_k x^{(k)} + g_k \quad (2.14)$$

$$k = 0, 1, \dots$$

- 3) если

$$\frac{\| \mathbf{x}^{(k+1)} - \mathbf{x}^{(k)} \|}{\| \mathbf{x}^{(k+1)} \|} \leq \varepsilon_p$$

или

$$\frac{\| \mathbf{r}^{(k+1)} \|}{\| \mathbf{x}^{(k+1)} \|} = \frac{\| \mathbf{A}\mathbf{x}^{(k+1)} - \mathbf{f} \|}{\| \mathbf{x}^{(k+1)} \|} \leq \varepsilon_p,$$

тогда итерационный процесс завершён и

$$\frac{\| \mathbf{x}^{(k+1)} - \mathbf{x}^* \|}{\| \mathbf{x}^* \|} \leq \varepsilon_p.$$

Итерационный процесс (2.13) называется стационарным, а процесс (2.14), соответственно, нестационарным. Как последовательные приближения $\mathbf{x}^{(k)}$ соотносятся с точным решением задачи (2.1)? Для того чтобы ответить на этот вопрос, проведем простой анализ. Каждое приближение $\mathbf{x}^{(k)}$ может быть представлено в следующем виде:

$$\begin{aligned} \mathbf{x}^{(k)} &= \mathbf{x}^* + \mathbf{e}^{(k)} \\ \mathbf{x}^{(k+1)} &= \mathbf{x}^* + \mathbf{e}^{(k+1)}, \end{aligned} \quad (2.15)$$

где $\mathbf{e}^{(k)}$ - вектор ошибки k -го приближения. Подставив (2.15) в (2.13) и учитывая (2.12), мы получим

$$\mathbf{e}^{(k+1)} = \mathbf{B}\mathbf{e}^{(k)} \text{ или } \mathbf{e}^{(k)} = \mathbf{B}^k \mathbf{e}^{(0)}, k = 0, 1, \dots \quad (2.16)$$

Если

$$\lim_{k \rightarrow \infty} \|\mathbf{e}^{(k)}\| = 0,$$

тогда итерационный процесс сходится и

$$\lim_{k \rightarrow \infty} \mathbf{x}^{(k)} = \mathbf{x}^*.$$

Разложим вектор $\mathbf{e}^{(0)}$ по собственным векторам матрицы $\mathbf{B}$:

$$\mathbf{e}^{(0)} = \sum_{n=1}^N \alpha_n \mathbf{z}_n, \quad \|\mathbf{z}_n\| = 1.$$

Тогда

$$\mathbf{e}^{(k)} = \sum_{n=1}^N \alpha_n \mathbf{B}^k \mathbf{z}_n = \sum_{n=1}^N \alpha_n \lambda_n^k(\mathbf{B}) \mathbf{z}_n$$

Оценка нормы вектора $\mathbf{e}^{(k)}$ даёт

$$\|\mathbf{e}^{(k)}\| \leq \sum_{n=1}^N |\alpha_n| \cdot |\lambda_n(\mathbf{B})|^k$$

Следовательно

$$\lim_{k \rightarrow \infty} \|\mathbf{e}^{(k)}\| = 0,$$

если все $|\lambda_n(\mathbf{B})| < 1$. Этот простой анализ поясняет основную теорему итерационных методов.

Теорема 2.2

Итерационный процесс (2.13) сходится для любого начального вектора $\mathbf{x}^{(0)}$ тогда и только тогда, когда

$$s(\mathbf{B}) < 1, \quad (2.17)$$

здесь $s(\mathbf{B}) = \max_n |\lambda_n(\mathbf{B})|$ - спектральный радиус матрицы $\mathbf{B}$.

Достаточное условие сходимости итерационного процесса:

$$\|\mathbf{B}\| < 1,$$

тогда условие (2.17) также выполняется, потому что $s(\mathbf{B}) \leq \|\mathbf{B}\|$ для любой матричной нормы.

На результат итерационного процесса (2.13) влияют ошибка алгоритма и, естественно, ошибки округления. Ошибку алгоритма можно легко оценить. Пусть, для простоты, итерации начинаются с вектора $\mathbf{x}^{(0)} = \mathbf{g}$, тогда

$$\mathbf{x}^{(k)} = \sum_{l=0}^k \mathbf{B}^l \mathbf{g}, \quad \mathbf{x}^* = \sum_{l=0}^{\infty} \mathbf{B}^l \mathbf{g}$$

Следовательно

$$\|\mathbf{e}_a^{(k)}\| = \|\mathbf{x}^* - \mathbf{x}^{(k)}\| = \left\| \sum_{l=k+1}^{\infty} \mathbf{B}^l \mathbf{g} \right\| \leq \sum_{l=k+1}^{\infty} \|\mathbf{B}\|^l \cdot \|\mathbf{g}\|.$$

Для $\|B\| < 1$ мы получаем

$$\|e_a^{(k)}\| \leq \frac{\|B\|^{k+1} \cdot \|g\|}{1 - \|B\|},$$

таким образом, чем меньше спектральный радиус матрицы B , тем быстрее приближения $x^{(k)}$ стремятся к x . Что касается ошибок округления, то следующая оценка может быть получена:

$$\|e_r^{(k)}\| \leq \frac{\varepsilon_1 \|g\|}{1 - \|B\|}, \text{ для всех } k.$$

Эта оценка показывает, что при использовании итерационных методов ошибки округления не накапливаются.

2.7.2 Метод Якоби.

Представим матрицу A в следующей форме:

$$A = L + D + U, \quad (2.18)$$

где

$$L = \begin{pmatrix} 0 & & & & \\ * & . & & & 0 \\ * & * & . & & \\ * & . & * & . & \\ * & * & * & * & 0 \end{pmatrix},$$

$$D = \text{diag}(a_{11}, \dots, a_{nn}),$$

$$U = \begin{pmatrix} 0 & * & * & * & * \\ . & & * & . & * \\ . & . & * & * & \\ 0 & . & . & * & \\ & & & 0 \end{pmatrix}.$$

Один из простейших способов построения эквивалентной формы (2.12) следующий:

$$\mathbf{Ax} = (\mathbf{L} + \mathbf{D} + \mathbf{U})\mathbf{x} = \mathbf{f}$$

$$\mathbf{Dx} = -(\mathbf{L} + \mathbf{U})\mathbf{x} + \mathbf{f} = (\mathbf{D} - \mathbf{A})\mathbf{x} + \mathbf{f},$$

и окончательно

$$\mathbf{x} = (\mathbf{I} - \mathbf{D}^{-1}\mathbf{A})\mathbf{x} + \mathbf{D}^{-1}\mathbf{f} \quad (2.19)$$

Тогда итерационная матрица $\mathbf{B}$ и вектор $\mathbf{g}$ для метода Якоби записываются как

$$\mathbf{B} = \mathbf{B}_J = \mathbf{I} - \mathbf{D}^{-1}\mathbf{A}$$

$$\mathbf{g} = \mathbf{g}_J = \mathbf{D}^{-1}\mathbf{f}$$

Построение эквивалентной формы (2.19) состоит в явном выражении n -го неизвестного из n -го уравнения системы. Так как матрица $\mathbf{B}_J$ легко вычисляется, можно оценить $s(\mathbf{B}_J)$ и определить сходится ли метод Якоби. Имеются также достаточные условия, которые позволяют выяснить сходимость этого метода непосредственно через компоненты матрицы $\mathbf{A}$. Итерационный процесс (2.13) с матрицей $\mathbf{B} = \mathbf{B}_J$ и вектором $\mathbf{g} = \mathbf{g}_J$ сходится, если

- 1) $\mathbf{A}$ есть матрица со строгим диагональным преобладанием. В этом случае, учитывая структуру матрицы $\mathbf{B}_J$, можно легко показать, что $\|\mathbf{B}_J\|_\infty < 1$,
- 2) $\mathbf{A}$ есть неразложимая матрица со слабым диагональным преобладанием.

Пример 2.2 (метод Якоби)

Рассмотрим систему (2.1), где матрица $\mathbf{A}$ и вектор $\mathbf{f}$ имеют следующий вид:

$$\mathbf{A} = \begin{pmatrix} 2 & -1 & 0 \\ -1 & 2 & -1 \\ 0 & -1 & 2 \end{pmatrix}, \quad \mathbf{f} = \begin{pmatrix} 1/3 \\ 1 \\ -1/3 \end{pmatrix} \quad (2.20)$$

Эта система имеет точное решение $\mathbf{x}^* = (2/3, 1, 1/3)^T$. Итерационная матрица $\mathbf{B}_J$ и вектор $\mathbf{g}_J$ записываются как

$$\mathbf{B}_J = \begin{pmatrix} 0 & 0.5 & 0 \\ 0.5 & 0 & 0.5 \\ 0 & 0.5 & 0 \end{pmatrix}, \quad \mathbf{g}_J = \begin{pmatrix} 1/6 \\ 1/2 \\ -1/6 \end{pmatrix}$$

Собственные значения матрицы $\mathbf{B}_J$:

$$\lambda(\mathbf{B}_J) = -0.5\sqrt{2}, 0, 0.5\sqrt{2}, \quad s(\mathbf{B}_J) = 0.5\sqrt{2} \approx 0.7071$$

Следовательно, метод Якоби для данной системы уравнений сходится. Результаты вычислений для начального вектора $\mathbf{x}^{(0)} = (0, 0, 0)^T$ и $\varepsilon_p = 10^{-5}$ представлены в Таблице 2.1.

Таблица 2.1

k	$\mathbf{x}^{(k)}$	$\ \mathbf{x}^{(k)} - \mathbf{x}^{(k-1)}\ _2$	$\ \mathbf{e}^{(k)}\ _2$
1	0.1666 0.5000 -0.1666	5.5277e-01	8.5602e-01
2	0.4166 0.5000 0.0833	3.5355e-01	6.1237e-01
.	...	...	...
10	0.6510 0.9687 0.3177	2.2097e-02	3.8272e-02
.	...	...	...
20	0.6661 0.9990 0.3328	6.9053e-04	1.1956e-03
.	...	...	...
33	0.6666 0.9999 0.3333	7.6293e-06	1.2619e-05

2.7.3 Метод Гаусса-Зейделя.

Обратимся опять к выражению (2.18). Только теперь мы построим эквивалентную форму (2.12) в другом виде:

$$\mathbf{Ax} = (\mathbf{L} + \mathbf{D} + \mathbf{U})\mathbf{x} = \mathbf{f},$$

$$(\mathbf{L} + \mathbf{D})\mathbf{x} = -\mathbf{U}\mathbf{x} + \mathbf{f},$$

$$(\mathbf{B} = \mathbf{B}_{GS} = -(\mathbf{L} + \mathbf{D})^{-1}\mathbf{U}, \mathbf{g} = \mathbf{g}_{GS} = -(\mathbf{L} + \mathbf{D})^{-1}\mathbf{f})$$

или

$$(\mathbf{U} + \mathbf{D})\mathbf{x} = -\mathbf{L}\mathbf{x} + \mathbf{f},$$

$$(\mathbf{B} = \mathbf{B}_{GS} = -(\mathbf{U} + \mathbf{D})^{-1}\mathbf{L}, \mathbf{g} = \mathbf{g}_{GS} = -(\mathbf{U} + \mathbf{D})^{-1}\mathbf{f})$$

Обращать матрицу $\mathbf{L} + \mathbf{D}$ или $\mathbf{U} + \mathbf{D}$ не имеет смысла, так как мы можем построить итерационную схему в виде

$$(\mathbf{L} + \mathbf{D})\mathbf{x}^{(k+1)} = -\mathbf{U}\mathbf{x}^{(k)} + \mathbf{f} \quad (2.21)$$

или

$$(\mathbf{U} + \mathbf{D})\mathbf{x}^{(k+1)} = -\mathbf{L}\mathbf{x}^{(k)} + \mathbf{f} \quad (2.22)$$

Матрица $\mathbf{L} + \mathbf{D}$ нижняя треугольная, а матрица $\mathbf{U} + \mathbf{D}$ верхняя треугольная, поэтому системы (2.21) и (2.22) могут быть легко решены относительно вектора $\mathbf{x}^{(k+1)}$. Так как матрица $\mathbf{B}_{GS}$ в явном виде не вычисляется и анализ её собственных значений невозможен, то метод Гаусса-Зейделя обычно применяется к системам, удовлетворяющим определённым условиям, а именно:

- 1) $\mathbf{A}$ есть матрица со строгим диагональным преобладанием,
- 2) $\mathbf{A}$ есть неразложимая матрица со слабым диагональным преобладанием.
- 3) $\mathbf{A}$ есть SPD матрица. В этом случае, если метод Якоби сходится ($s(\mathbf{B}_J) < 1$), тогда метод Гаусса-Зейделя сходится более быстро, так как $s(\mathbf{B}_{GS}) = s^2(\mathbf{B}_J)$ для матриц этого типа.

Пример 2.3 (метод Гаусса-Зейделя)

Рассмотрим систему (2.20) из примера 2.2. Итерационная матрица $\mathbf{B}_{GS}$ и вектор $\mathbf{g}_{GS}$ для этой системы имеют следующий вид

$$\mathbf{B}_{GS} = \begin{pmatrix} 0 & 0.5 & 0 \\ 0 & 0.25 & 0.5 \\ 0 & 0.125 & 0.25 \end{pmatrix}, \mathbf{g}_{GS} = \begin{pmatrix} 1/6 \\ 7/12 \\ 1/8 \end{pmatrix}$$

Собственные значения матрицы $\mathbf{B}_{GS}$:

$$\lambda(\mathbf{B}_{GS}) = 0, 0, 0.5, s(\mathbf{B}_{GS}) = 0.5.$$

Результаты вычислений для начального вектора $\mathbf{x}^{(0)} = (0, 0, 0)^T$ и $\epsilon_p = 10^{-5}$ представлены в Таблице 2.2.

Таблица 2.2

k	$\mathbf{x}^{(k)}$	$\ \mathbf{x}^{(k)} - \mathbf{x}^{(k-1)}\ _2$	$\ \mathbf{e}^{(k)}\ _2$
1	0.1666	6.1941e-01	6.8338e-01
	0.5833		
	0.1250		
2	0.4583	3.7325e-01	3.1249e-01
	0.7916		
	0.2291		
.	...	...	...
10	0.6658	1.2207e-03	1.2200e-03
	0.9991		
	0.3329		
.	...	...	...
17	0.6666	9.5169e-06	8.9655e-06
	0.9999		
	0.3333		

Как показывают эти результаты, для данной системы метод Гаусса-Зейделя сходится быстрее, чем метод Якоби. Это объясняется тем, что матрица $\mathbf{A}$ определённая в (2.20) есть SPD матрица и $s(\mathbf{B}_{GS}) < s(\mathbf{B}_J)$.

2.7.4 Метод релаксации.

Основной вопрос, возникающий при использовании итерационных методов: как достичь наиболее быстрой сходимости, или другими словами, как построить итерационную матрицу для данной системы,

которая имела бы наименьший спектральный радиус $s(\mathbf{B})$. Для этого необходимо иметь некоторый способ управления свойствами итерационной матрицы $\mathbf{B}$. Один из таких способов реализуется в методе релаксации.

В начале обсудим общую схему этого метода. Рассмотрим систему (2.1) с SPD матрицей $\mathbf{A}$. Для вектора приближенного решения $\mathbf{x}^{(k)}$ мы определим функционал ошибки в виде

$$\varphi(\mathbf{x}^{(k)}) = (\mathbf{x}^{(k)} - \mathbf{x}^*, \mathbf{A}(\mathbf{x}^{(k)} - \mathbf{x}^*)) = (\mathbf{e}^{(k)}, \mathbf{A}\mathbf{e}^{(k)}) = (\mathbf{e}^{(k)}, \mathbf{r}^{(k)}).$$

Для того чтобы вычислить другой вектор $\mathbf{x}^{(k+1)}$, который отличается от $\mathbf{x}^{(k)}$ только n -ой компонентой и такой, что $\varphi(\mathbf{x}^{(k+1)}) < \varphi(\mathbf{x}^{(k)})$, положим $\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + \alpha \mathbf{b}_n$ ($\mathbf{b}_n$ есть базисный вектор, смотри 2.1). Тогда легко получить, что

$$\varphi(\mathbf{x}^{(k+1)}) = \varphi(\mathbf{x}^{(k)}) + \frac{(\alpha a_{nn} + r_n^{(k)})^2}{a_{nn}} - \frac{(r_n^{(k)})^2}{a_{nn}}, \quad (2.23)$$

где

$$r_n^{(k)} = (\mathbf{A}\mathbf{x}^{(k)} - \mathbf{f}, \mathbf{b}_n)$$

есть n -ая компонента вектора невязки. Сумма второго и третьего слагаемых в (2.23) принимает отрицательное значение, если выбрать параметр α так, чтобы

$$|\alpha a_{nn} + r_n^{(k)}| < |r_n^{(k)}|$$

или

$$\alpha = -\omega \frac{r_n^{(k)}}{a_{nn}}, \quad 0 < \omega < 2.$$

Тогда n -ая компонента вектора $\mathbf{x}^{(k+1)}$ вычисляется как

$$x_n^{(k+1)} = x_n^{(k)} - \omega \frac{r_n^{(k)}}{a_{nn}},$$

где ω называется параметром релаксации. Выбирая таким же образом параметр α для $n=1, \dots, N$, мы получим следующую итерационную схему

$$\begin{aligned} \mathbf{x}^{(k+1)} &= \mathbf{x}^{(k)} - \omega \mathbf{D}^{-1}(\mathbf{A}\mathbf{x}^{(k)} - \mathbf{f}) = \\ &= ((1-\omega)\mathbf{I} + \omega \mathbf{B}_J)\mathbf{x}^{(k)} + \omega \mathbf{D}^{-1}\mathbf{f} = \mathbf{B}_{JR}(\omega)\mathbf{x}^{(k)} + \mathbf{g}_{JR}(\omega) \end{aligned} \quad (2.24)$$

Эквивалентная форма (2.12) для этой итерационной схемы может быть также записана в виде

$$\mathbf{D}\mathbf{x} = \mathbf{D}\mathbf{x} - \omega(\mathbf{A}\mathbf{x} - \mathbf{f}). \quad (2.25)$$

Теперь мы имеем итерационную матрицу, которая зависит от параметра ω . Можно подобрать такое значение параметра ω , которое обеспечивает минимальное значение $s(\mathbf{B}_{JR}(\omega))$. Такое значение называется оптимальным значением параметра релаксации и обозначается ω_{opt} . Метод релаксации (2.24) основывается на методе Якоби ($\mathbf{B}_{JR}(1)=\mathbf{B}_J$) и $\lambda(\mathbf{B}_{JR}(\omega))=1+\omega(\lambda(\mathbf{B}_J)-1)$. Пусть $\lambda_l \leq \lambda(\mathbf{B}_J) \leq \lambda_r$, тогда оптимальное значение параметра релаксации вычисляется как

$$\omega_{\text{opt}} = \frac{2}{2 + \lambda_l - \lambda_r}$$

и

$$s(\mathbf{B}_{JR}(\omega_{\text{opt}})) = \frac{\lambda_l + \lambda_r}{2 + \lambda_l - \lambda_r}$$

Пример 2.4 (вычисление оптимального параметра релаксации для итерационной схемы (2.24))

Рассмотрим следующую матрицу для системы (2.1):

$$\mathbf{A} = \begin{pmatrix} 3 & 1 & 1 \\ 1 & 3 & 1 \\ 1 & 1 & 3 \end{pmatrix}$$

Собственные значения матрицы Якоби $\mathbf{B}_J$: $-2/3$, $1/3$, $1/3$ и, следовательно, $s(\mathbf{B}_J)=2/3$. Учитывая, что $\lambda_l=2/3$ и $\lambda_r=1/3$, получим $\omega_{\text{opt}} \approx 0.8571$. Тогда $s(\mathbf{B}_{JR}(\omega_{\text{opt}})) \approx 0.4286 < s(\mathbf{B}_J)$ и это говорит о том, что можно ускорить сходимость по сравнению с методом Якоби.

Следующий логический шаг – построить метод релаксации основанный на методе Гаусса-Зейделя (2.21). Учитывая выражение (2.18), можно преобразовать (2.25) в следующую эквивалентную систему:

$$\mathbf{D}\mathbf{x} = \mathbf{D}\mathbf{x} - \omega(\mathbf{L} + \mathbf{D} + \mathbf{U})\mathbf{x} + \omega\mathbf{f},$$

$$\begin{aligned}
 (\mathbf{D} + \omega \mathbf{L})\mathbf{x} &= (\mathbf{D} - \omega(\mathbf{D} + \mathbf{U}))\mathbf{x} + \omega \mathbf{f}, \\
 (\mathbf{B}_{\text{SOR}} &= (\mathbf{D} + \omega \mathbf{L})^{-1}(\mathbf{D} - \omega(\mathbf{D} + \mathbf{U}))).
 \end{aligned}$$

Как и прежде, обращаться матрицу $\mathbf{D} + \omega \mathbf{L}$ не нужно, а записать итерационную схему в виде

$$(\mathbf{D} + \omega \mathbf{L})\mathbf{x}^{(k+1)} = (\mathbf{D} - \omega(\mathbf{D} + \mathbf{U}))\mathbf{x}^{(k)} + \omega \mathbf{f}, \quad k = 0, \dots \quad (2.26)$$

Матрица $\mathbf{D} + \omega \mathbf{L}$ нижняя треугольная и система (2.26) может быть легко решена относительно вектора $\mathbf{x}^{(k+1)}$. На практике было замечено (в некоторых случаях доказано), что итерационная схема (2.26) сходится медленнее при $\omega < 1$ (нижняя релаксация), чем при $\omega \geq 1$ (верхняя релаксация). Поэтому обычно используется $\omega \geq 1$ и за итерационной схемой (2.26) закрепилось название метод последовательной верхней релаксации (SOR, Successive Over-Relaxation). Для небольшого числа специфических (но важных) задач оптимальное значение параметра релаксации может быть определено. Рассмотрим кратко этот случай.

Матрица Якоби

$$\mathbf{B}_j = \mathbf{I} - \mathbf{D}^{-1}\mathbf{A} = -\mathbf{D}^{-1}\mathbf{L} - \mathbf{D}^{-1}\mathbf{U}$$

называется согласованно упорядоченной, если

$$\lambda(-\alpha \mathbf{D}^{-1}\mathbf{L} - \frac{1}{\alpha} \mathbf{D}^{-1}\mathbf{U}) = \lambda(\mathbf{B}_j)$$

для любого параметра $\alpha \neq 0$. Например, блочно-трехдиагональные матрицы вида

$$\mathbf{T} = \begin{pmatrix}
 \mathbf{D}_1 & \mathbf{U}_1 & & & \\
 \mathbf{L}_2 & \mathbf{D}_2 & \mathbf{U}_2 & & 0 \\
 & \cdot & \cdot & \cdot & \\
 & & \cdot & \cdot & \cdot \\
 0 & & & \mathbf{L}_{p-1} & \mathbf{D}_{p-1} & \mathbf{U}_{p-1} \\
 & & & & \mathbf{L}_p & \mathbf{D}_p
 \end{pmatrix},$$

где $\mathbf{D}_m$ - квадратная матрица размером $N_m \times N_m$ ($m=1, \dots, p$), $\mathbf{L}_m$ и $\mathbf{U}_{m-1}$ - матрицы размером $N_m \times N_{m+1}$ и $N_{m-1} \times N_m$ соответственно ($m=2, \dots, p$; $N_1 + \dots + N_p = N$) являются согласованно упорядоченными. Пусть $\mathbf{A}$ есть SPD согласованно упорядоченная матрица. Если метод Якоби для этой системы сходится ($s(\mathbf{B}_J) < 1$), то значение оптимального параметра релаксации для метода SOR вычисляется по формуле:

$$\omega_{\text{opt}} = \frac{2}{1 + \sqrt{1 - s^2(\mathbf{B}_J)}} \quad \text{и} \quad s(\mathbf{B}_{\text{SOR}}) = \omega_{\text{opt}} - 1 \quad (2.27)$$

Пример 2.5 (метод SOR)

Рассмотрим снова систему (2.20) из примера 2.2. Матрица $\mathbf{A}$ трехдиагональная (это специальный случай блочно-трехдиагональной матрицы) и, следовательно, согласованно упорядоченная. В дополнение, она также является SPD матрицей, и мы можем вычислить ω_{opt} согласно выражению (2.27). Для данной матрицы

$$s(\mathbf{B}_J) = 0.5\sqrt{2},$$

тогда $\omega_{\text{opt}} \approx 1.17157$ и $s(\mathbf{B}_{\text{SOR}}) \approx 0.17157$. Это означает, что для данной системы метод SOR обеспечивает очень быструю сходимость по сравнению с методами рассмотренными выше. Результаты вычислений для начального вектора $\mathbf{x}^{(0)} = (0, 0, 0)^T$ и $\varepsilon_p = 10^{-5}$ представлены в Таблице 2.3.

Таблица 2.3

k	$\mathbf{x}^{(k)}$	$\ \mathbf{x}^{(k)} - \mathbf{x}^{(k-1)}\ _2$	$\ \mathbf{e}^{(k)}\ _2$
1	0.1952 0.7001 0.2148	7.5798e-01	5.7109e-01
2	0.5719 0.9265 0.3106	4.4975e-01	1.2202e-01
...	...	...	...
9	0.6666 0.9999 0.3333	7.9500e-06	1.5046e-06

Как показывают эти результаты, требуется только 9 итераций, для того чтобы получить приближенное решение с заданной точностью и это в два раза меньше по сравнению с методом Гаусса-Зейделя.

Вычисление спектрального радиуса матрицы Якоби часто требует значительных вычислительных затрат, поэтому иногда прибегают к грубой оценке $s(\mathbf{B}_J)$, что позволяет получить приемлемое значение параметра релаксации, близкое к ω_{opt} .

Что касается других типов матриц, то простые способы вычисления значения оптимального параметра релаксации не известны. Однако, известны достаточные условия при которых метод SOR сходится:

- 1) если $\mathbf{A}$ есть SPD матрица, то метод SOR сходится при $0 < \omega < 2$. В этом случае, значение параметра релаксации, определяемого по формуле (2.27), близко к оптимальному, так как

$$\omega_{\text{opt}} - 1 \leq s(\mathbf{B}_{\text{SOR}}) \leq \sqrt{\omega_{\text{opt}} - 1}.$$

- 2) если $\mathbf{A}$ есть матрица со строгим диагональным преобладанием или неразложимая матрица со слабым диагональным преобладанием, то существует такое $\omega > 1$, что метод SOR сходится при $0 < \omega < \omega^*$.

2.7.5 Вариационно-итерационные методы.

Как следует из предыдущего рассмотрения, анализ сходимости итерационного метода предусматривает знание спектрального радиуса итерационной матрицы. Однако вычисление или оценка $s(\mathbf{B})$ часто требует неоправданно больших вычислительных затрат. Существуют методы, сходимость которых обеспечивается с помощью специальных параметров, которые, в свою очередь, зависят от приближённого решения. Очевидно, что осуществление такого подхода приводит к нестационарным итерационным процессам вида (2.14).

Мы рассмотрим итерационные схемы следующего вида:

$$\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} - \tau_k (\mathbf{A}\mathbf{x}^{(k)} - \mathbf{f}), \quad k = 0, 1, \dots \quad (2.28)$$

где $\tau_k \neq 0$ - итерационный параметр. Введём вектор невязки $\mathbf{r}^{(k)} = \mathbf{A}\mathbf{x}^{(k)} - \mathbf{f}$. Используя выражение (2.28), вектор невязки $\mathbf{r}^{(k+1)}$, связанный со следующим приближением, может быть выражен через $\mathbf{r}^{(k)}$:

$$\begin{aligned} \mathbf{Ax}^{(k+1)} - \mathbf{f} &= (\mathbf{Ax}^{(k)} - \mathbf{f}) - \tau_k \mathbf{A}(\mathbf{Ax}^{(k)} - \mathbf{f}) = \\ \mathbf{r}^{(k)} - \tau_k \mathbf{Ar}^{(k)} &= \mathbf{r}^{(k+1)} \end{aligned}$$

Тогда мы можем определить параметр τ_k таким образом, чтобы $\|\mathbf{r}^{(k+1)}\|_2$ была минимальной. Норма вектора невязки $\mathbf{r}^{(k+1)}$ выражается следующим образом

$$\begin{aligned} \|\mathbf{r}^{(k+1)}\|_2^2 &= (\mathbf{r}^{(k+1)}, \mathbf{r}^{(k+1)}) = (\mathbf{r}^{(k)} - \tau_k \mathbf{Ar}^{(k)}, \mathbf{r}^{(k)} - \tau_k \mathbf{Ar}^{(k)}) = \\ &= (\mathbf{r}^{(k)}, \mathbf{r}^{(k)}) - 2\tau_k (\mathbf{r}^{(k)}, \mathbf{Ar}^{(k)}) + \tau_k^2 (\mathbf{Ar}^{(k)}, \mathbf{Ar}^{(k)}) = \varphi(\tau_k) \end{aligned} \quad (2.29)$$

Условие $\varphi'(\tau_k)=0$ есть условие минимума функции $\varphi(\tau_k)$, так как

$$\varphi''(\tau_k) = (\mathbf{Ar}^{(k)}, \mathbf{Ar}^{(k)}) = \|\mathbf{Ar}^{(k)}\|_2^2 > 0.$$

Тогда итерационный параметр τ_k вычисляется по формуле

$$\tau_k = \frac{(\mathbf{r}^{(k)}, \mathbf{Ar}^{(k)})}{(\mathbf{Ar}^{(k)}, \mathbf{Ar}^{(k)})}. \quad (2.30)$$

Итерационная схема (2.28) и (2.30) называется методом минимальных невязок. Проанализируем сходимость этого метода. Если мы подставим выражение (2.30) в (2.29), то получим

$$\|\mathbf{r}^{(k+1)}\|_2^2 = \|\mathbf{r}^{(k)}\|_2^2 - \frac{(\mathbf{r}^{(k)}, \mathbf{Ar}^{(k)})^2}{(\mathbf{Ar}^{(k)}, \mathbf{Ar}^{(k)})} > 0. \quad (2.31)$$

Оба слагаемых в правой части выражения (2.31) положительны, следовательно,

$$\|\mathbf{r}^{(k+1)}\|_2^2 < \|\mathbf{r}^{(k)}\|_2^2$$

для любой определённой матрицы $\mathbf{A}$ и это доказывает сходимость итерационной схемы (2.28). Теперь рассмотрим скорость сходимости метода минимальных невязок. Предположим, что матрица $\mathbf{A}$ системы (2.1) имеет следующие свойства:

$$\gamma_1 \leq \frac{(\mathbf{x}, \mathbf{Ax})}{(\mathbf{x}, \mathbf{x})} \leq \gamma_2, \quad \mathbf{x} \in \mathbb{R}^n \quad (0 < \gamma_1 \leq \gamma_2 < \infty)$$

и

$$\| \mathbf{A} - \mathbf{A}^T \| \leq 2\gamma_3.$$

Тогда $\| \mathbf{r}^{(k+1)} \|_2 = q \| \mathbf{r}^{(k)} \|_2$ где

$$q = \frac{\gamma_2^2 - \gamma_1^2 + 4\gamma_3 \sqrt{\gamma_3^2 + \gamma_1 \gamma_2}}{(\gamma_1 + \gamma_2)^2 + 4\gamma_3^2} < 1.$$

Если матрица $\mathbf{A}$ имеет вещественные собственные значения, то можно положить

$$\gamma_2 = \max_n \lambda_n(\mathbf{A}) \text{ и } \gamma_1 = \min_n \lambda_n(\mathbf{A}).$$

Пример 2.6 (скорость сходимости метода минимальных невязок)

Рассмотрим матрицу $\mathbf{A}$ из примера 2.2. Эта матрица симметричная ($\gamma_3=0$) и

$$\gamma_2 = \max_n \lambda_n(\mathbf{A}) = 2 + \sqrt{2}, \quad \gamma_1 = \min_n \lambda_n(\mathbf{A}) = 2 - \sqrt{2}.$$

Тогда величина

$$q = 0.5\sqrt{2} \approx 0.7071,$$

что говорит о довольно медленной сходимости.

Для SPD матриц можно также использовать итерационную схему (2.28), только вычислять параметр τ_k другим способом. Вместо вектора невязки возьмём вектор ошибки $\mathbf{e}^{(k)} = \mathbf{x}^{(k)} - \mathbf{x}^*$. Учитывая, что $\mathbf{A}\mathbf{x} = \mathbf{f}$, из выражения (2.28) следует

$$\begin{aligned} \mathbf{e}^{(k+1)} &= \mathbf{x}^{(k+1)} - \mathbf{x}^* = \mathbf{x}^{(k)} - \mathbf{x}^* + \tau_k \mathbf{A}(\mathbf{x}^{(k)} - \mathbf{x}^*) = \\ &= \mathbf{e}^{(k)} - \tau_k \mathbf{A} \mathbf{e}^{(k)} \end{aligned} \quad (2.32)$$

Мы не можем вычислить вектор ошибки, но

$$\mathbf{A} \mathbf{e}^{(k+1)} = \mathbf{A} \mathbf{x}^{(k)} - \mathbf{A} \mathbf{x}^* = \mathbf{A} \mathbf{x}^{(k)} - \mathbf{f} = \mathbf{r}^{(k)}.$$

Таким образом, можно определить параметр τ_k , минимизируя величину $(\mathbf{e}^{(k+1)}, \mathbf{A} \mathbf{e}^{(k+1)})$. Используя выражение (2.32), получим

$$(\mathbf{e}^{(k+1)}, \mathbf{A}\mathbf{e}^{(k+1)}) = (\mathbf{e}^{(k)} - \tau_k \mathbf{r}^{(k)}, \mathbf{r}^{(k)} - \tau_k \mathbf{A}\mathbf{r}^{(k)}) = \\ (\mathbf{e}^{(k)}, \mathbf{r}^{(k)}) - 2\tau_k (\mathbf{r}^{(k)}, \mathbf{r}^{(k)}) + \tau_k^2 (\mathbf{r}^{(k)}, \mathbf{A}\mathbf{r}^{(k)}) = \varphi(\tau_k)$$

Условие $\varphi'(\tau_k)=0$ есть условие минимума функции $\varphi(\tau_k)$, так как

$$\varphi''(\tau_k) = (\mathbf{r}^{(k)}, \mathbf{A}\mathbf{r}^{(k)}) > 0$$

($\mathbf{A}$ есть SPD матрица). Тогда параметр τ_k вычисляется по формуле

$$\tau_k = \frac{(\mathbf{r}^{(k)}, \mathbf{r}^{(k)})}{(\mathbf{r}^{(k)}, \mathbf{A}\mathbf{r}^{(k)})}. \quad (2.33)$$

Итерационная схема (2.28) с параметром τ_k определяемым выражением (2.33) называется методом наискорейшего спуска. По свойствам этот метод близок к методу минимальных невязок.

На основе метода наискорейшего спуска можно построить итерационную схему с более высокой скоростью сходимости. Для этого модифицируем выражение (2.28) и запишем его в виде:

$$\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} - \tau_k \mathbf{d}^{(k)}, \quad k = 0, 1, \dots \quad (2.34)$$

где $\mathbf{d}^{(k)}$ - вектор, который направлен к решению системы линейных уравнений (вектор направления). В методе наискорейшего спуска, вектор невязки $\mathbf{r}^{(k)}$ выбирается в качестве вектора направления, но для многих систем такой выбор оказывается не очень хорошим с точки зрения скорости сходимости. Лучшие вектора направления задаются следующими соотношениями

$$\mathbf{d}^{(0)} = \mathbf{r}^{(0)} \\ \mathbf{d}^{(k)} = \mathbf{r}^{(k)} + \alpha_k \mathbf{d}^{(k-1)}, \quad k = 1, 2, \dots \quad (2.35)$$

Параметр α_k определяется таким образом, чтобы вектора $\mathbf{d}^{(k)}$ и $\mathbf{A}\mathbf{d}^{(k-1)}$ были ортогональны $((\mathbf{d}^{(k)}, \mathbf{A}\mathbf{d}^{(k-1)}))$, тогда

$$\alpha_k = -\frac{(\mathbf{r}^{(k)}, \mathbf{A}\mathbf{d}^{(k-1)})}{(\mathbf{d}^{(k-1)}, \mathbf{A}\mathbf{d}^{(k-1)})} \quad (2.36)$$

Используя тот же самый подход, что и методе наискорейшего спуска, параметр τ_k определяется из условия минимума величины $(\mathbf{e}^{(k+1)}, \mathbf{A}\mathbf{e}^{(k+1)})$, что приводит к следующей формуле

$$\tau_k = \frac{(\mathbf{r}^{(k)}, \mathbf{d}^{(k)})}{(\mathbf{d}^{(k)}, \mathbf{A}\mathbf{d}^{(k)})} \quad (2.37)$$

Итерационная схема (2.34) вместе с (2.35), (2.36) и (2.37) называется методом сопряжённых градиентов. Ошибка k -го приближения, получаемого этим методом, подчиняется следующей оценке

$$(\mathbf{e}^{(k)}, \mathbf{A}\mathbf{e}^{(k)}) \leq 2(\mathbf{e}^{(0)}, \mathbf{A}\mathbf{e}^{(0)}) (q(\mathbf{A}))^k,$$

где

$$q(\mathbf{A}) = \frac{\sqrt{\max_n \lambda_n(\mathbf{A})} - \sqrt{\min_n \lambda_n(\mathbf{A})}}{\sqrt{\max_n \lambda_n(\mathbf{A})} + \sqrt{\min_n \lambda_n(\mathbf{A})}} < 1$$

Отличительной чертой этого метода является то, что, пренебрегая ошибками округлений, мы получаем $\mathbf{r}^{(k)}=0$ при некотором $k \leq N$, следовательно $\mathbf{x}^{(k)} = \mathbf{x}^*$. Это значит, что нужно не более чем N итераций, для того чтобы вычислить точное решение. Для больших систем уравнений это не очень утешительный результат. Тем не менее, во многих случаях метод сопряжённых градиентов и его модификации сходятся очень быстро.

Как мы видим, скорость сходимости метода сопряжённых градиентов зависит от спектра матрицы коэффициентов системы уравнений. Для того чтобы улучшить сходимость этого итерационного метода, можно попытаться преобразовать исходную систему в другую систему, имеющую то же самое решение и матрица коэффициентов которой имеет более узкий спектр. Такое преобразование осуществляется с помощью специальной матрицы, которая называется предобуславливателем. Хороший предобуславливатель значительно ускоряет сходимость, что более чем компенсирует вычислительные затраты необходимые для применения преобразования. Например, если матрица $\mathbf{P}$ близка к матрице $\mathbf{A}$, то преобразованная система

$$\mathbf{P}^{-1}\mathbf{A}\mathbf{x} = \mathbf{P}^{-1}\mathbf{f}$$

имеет тоже решение что и $\mathbf{A}\mathbf{x}=\mathbf{f}$, но собственные значения матрицы $\mathbf{P}^{-1}\mathbf{A}$ будут близки к единице (тогда $q(\mathbf{P}^{-1}\mathbf{A})$ будет мало).

В качестве простого предобуславливателя можно использовать диагональ исходной матрицы коэффициентов:

$$\mathbf{P} = \text{diag}(a_{11}, \dots, a_{nn}).$$

Преобразование, основанное на таком выборе матрицы $\mathbf{P}$, называется преобусловливанием Якоби. Применение такого преобусловливания осуществляется очень просто, хотя с другой стороны, применение более сложных процедур обычно даёт лучшие результаты.

Пример 2.7 (применение преобусловливания Якоби)

Рассмотрим систему уравнений с матрицей коэффициентов

$$\mathbf{A} = \begin{pmatrix} 2 & 1 & 0 \\ 1 & 4 & 1 \\ 0 & 1 & 2 \end{pmatrix}$$

Скорость сходимости метода сопряжённых градиентов для системы $\mathbf{Ax}=\mathbf{f}$ определяется параметром $q(\mathbf{A})\approx 0.315$. После применения преобусловливателя Якоби $\mathbf{P}=\text{diag}(2, 4, 2)$ к матрице $\mathbf{A}$, мы получим преобразованную матрицу

$$\mathbf{P}^{-1}\mathbf{A} = \begin{pmatrix} 0.5 & 0 & 0 \\ 0 & 0.25 & 0 \\ 0 & 0 & 0.5 \end{pmatrix} \begin{pmatrix} 2 & 1 & 0 \\ 1 & 4 & 1 \\ 0 & 1 & 2 \end{pmatrix} =$$

$$\begin{pmatrix} 1 & 0.5 & 0 \\ 0.25 & 1 & 0.25 \\ 0 & 0.5 & 1 \end{pmatrix}$$

Теперь скорость сходимости метода сопряжённых градиентов для системы $\mathbf{P}^{-1}\mathbf{Ax}=\mathbf{P}^{-1}\mathbf{f}$ определяется параметром $q(\mathbf{P}^{-1}\mathbf{A})\approx 0.128$. Таким образом, достигается значительное ускорение сходимости по сравнению с исходной системой.

2.8 Какие методы более эффективны: прямые или итерационные?

Ответить на этот вопрос в общем случае не возможно – это зависит от конкретной проблемы. Мы можем сделать только некоторые

приблизительные оценки. Пусть **A** является матрицей общего вида. Согласно выражению (2.13) необходимо $\sim 2N^2$ операций, для того чтобы вычислить следующее приближение к решению. Тогда K итераций потребуют приблизительно $2KN^2$ операций, и по сравнению с методом LU-разложения, вычислительные затраты для K итераций будут меньше, если

$$N^3 > 2KN^2 \text{ или } K < \frac{1}{2}N$$

Пусть теперь **A** является ленточной матрицей с шириной ленты m , и сама лента разрежена, скажем, она имеет p ненулевых элементов в строке. В этой ситуации, затраты для метода LU-разложения составляют m^2N операций, в то время как затраты для K итераций составляют приблизительно $2pKN$ операций. Поэтому итерационные методы должны сходиться меньше чем за $m^2/2p$ итераций, для того чтобы быть более эффективными. Например, для матриц возникающих при решении уравнений в частных производных m приблизительно равно $N^{1/2}$ и $p \geq 5$. Число уравнений в задачах этого типа может лежать в пределах от 10^2 до 10^6 . Следовательно, можно заключить, что итерационные методы более эффективны при решении больших систем линейных уравнений.

ГЛАВА 3

Вычисление собственных значений и векторов

Список обозначений

$\mathbf{A}=\{a_{nm}\}$	матрица (заглавные буквы, жирный шрифт)
$\mathbf{x}$	вектор (строчные буквы, жирный шрифт)
N	размер матрицы или вектора
$\ \cdot\ $	норма вектора или матрицы
$\mathbf{A}^T, \mathbf{x}^T$	транспонированная матрица или вектор
$(\mathbf{x}, \mathbf{y})=\mathbf{x}^T \mathbf{y}$	скалярное произведение векторов $\mathbf{x}$ и $\mathbf{y}$
$\det(\mathbf{A})$	определитель матрицы
$\mathbf{I}$	единичная матрица ($\mathbf{I} = \text{diag}(1, \dots, 1)$)
$\mathbf{A}^{-1}$	обратная матрица ($\mathbf{A}^{-1} \mathbf{A} = \mathbf{I}$)
$\mathbf{z}_n$	n -ый собственный вектор
$\lambda_n(\mathbf{A})$	n -ое собственное значение матрицы $\mathbf{A}$
ε_p	требуемая точность вычисления приближенного решения
$\mathbf{x}^{(k)}$	k -ое приближение к собственному вектору
$\rho(\mathbf{x}) = (\mathbf{x}, \mathbf{A}\mathbf{x})/(\mathbf{x}, \mathbf{x})$	отношение Рэля для матрицы $\mathbf{A}$
$\mathbf{r}(\mathbf{x}) = \mathbf{A}\mathbf{x} - \rho(\mathbf{x})\mathbf{x}$	вектор невязки для задачи на собственное значение

Необходимость определения собственных значений квадратной матрицы возникает при решении различных научных и инженерных задач. В качестве примера можно привести такие проблемы как вибрация конструкций, уровни энергии в квантовых системах, колебательные движения молекул, решение и анализ систем линейных дифференциальных уравнений. Задача на собственное значение является нелинейной задачей, поэтому вычисление собственных значений и векторов возможно лишь на основе той или иной итерационной процедуры. В этой главе рассматриваются два основных подхода для решения задачи на собственное значение: степенной метод

и QR метод. Мы ограничимся рассмотрением собственных значений вещественных матриц, хотя аналогичные подходы применимы и к комплексным матрицам. В начале, мы рассмотрим основные определения из линейной алгебры, относящиеся к задаче на собственное значение.

3.1 Основные сведения из линейной алгебры, относящиеся к задаче на собственное значение.

Общие сведения из линейной алгебры представлены в параграфе 2.1. Здесь мы рассмотрим некоторые специфические определения, касающиеся задачи на собственное значение. В общем случае, эта задача формулируется в виде

$$\mathbf{A}\mathbf{z}_n = \lambda_n \mathbf{M}\mathbf{z}_n, n = 1, \dots, N$$

где $\mathbf{A}$ и $\mathbf{M}$ – квадратные матрицы размером N на N и пара $(\lambda_n, \mathbf{z}_n)$ есть собственное значение и соответствующий собственный вектор. В частности, такая проблема возникает при анализе вибрации конструкций. Мы ограничимся рассмотрением более простой проблемы (стандартная задача на собственное значение):

$$\mathbf{A}\mathbf{z}_n = \lambda_n \mathbf{z}_n, n = 1, \dots, N \quad (3.1)$$

Матрица $\mathbf{A}$ имеет N собственных значений (набор всех значений называется спектром матрицы $\mathbf{A}$), и они определяются как корни характеристического полинома $\det(\mathbf{A} - \lambda \mathbf{I})$. Собственные вектора $\mathbf{z}_n$ ($n = 1, \dots, N$) образуют систему ортонормированных векторов:

$$(\mathbf{z}_n, \mathbf{z}_m) = \begin{cases} 1, n = m \\ 0, n \neq m \end{cases}$$

В общем случае, матрица $\mathbf{A}$ имеет комплексные собственные значения. Только матрицы, обладающие специальными свойствами, имеют только вещественные собственные значения. Такие матрицы называются Эрмитовыми. Пусть символ “Н” обозначает последовательное применение двух операций: комплексного сопряжения и транспонирования,

тогда матрица $\mathbf{A}$ является Эрмитовой если $\mathbf{A}=\mathbf{A}^H$. Вещественная симметричная матрица есть частный случай Эрмитовой матрицы.

В дальнейшем, для определённости, мы будем нумеровать собственные значения следующим образом:

$$|\lambda_1| \leq |\lambda_2| \leq \dots \leq |\lambda_{N-1}| \leq |\lambda_N|.$$

Это значит, что λ_1 - минимальное по модулю собственное значение, а λ_N - максимальное по модулю собственное значение.

Отношение Рэлея для симметричной матрицы $\mathbf{A}$ определяется как

$$\rho(\mathbf{x}) = \frac{(\mathbf{x}, \mathbf{A}\mathbf{x})}{(\mathbf{x}, \mathbf{x})}$$

для любого вектора $\mathbf{x} \neq \mathbf{0}$. Оно имеет очень полезные свойства, а именно

- 1) $\lambda_1 \leq \rho(\mathbf{x}) \leq \lambda_N$ для всех ненулевых векторов $\mathbf{x} \in \mathbb{R}^N$
- 2) если $\mathbf{x} = \mathbf{z}_n$, тогда $\rho(\mathbf{x}) = \lambda_n$
- 3) на основе предыдущего свойства, вектор невязки для задачи на собственное значение может быть определён как

$$\mathbf{r}(\mathbf{x}) = \mathbf{A}\mathbf{x} - \rho(\mathbf{x})\mathbf{x}$$

и неравенство $\|\mathbf{r}(\mathbf{x})\| \leq \|\mathbf{A}\mathbf{x} - \alpha\mathbf{x}\|$ справедливо для любого числа α . Следовательно, если вектор $\mathbf{x}$ есть некоторое приближение к $\mathbf{z}_n$, то $\rho(\mathbf{x})$ есть наилучшее приближение к λ_n .

3.2 Локализация собственных значений.

Информация о расположении собственных значений на комплексной плоскости может быть получена без особых вычислительных затрат. Один из простых методов локализации собственных значений основан на следующей теореме.

Теорема 3.1 (Гершгорин)

Любое собственное значение матрицы $\mathbf{A}=\{a_{nm}\}$ лежит, по крайней мере, в одном из кругов

$$G_n = \left\{ \lambda : |\lambda - a_{nn}| \leq \sum_{\substack{m=1 \\ m \neq n}}^N |a_{nm}| = \text{sum}_n \right\} \quad (3.2)$$

Множество G_n , определяемое выражением (3.2) называется n -ым кругом Гершгорина матрицы A . Когда собственные значения вещественны, тогда круги Гершгорина это просто интервалы. Если n кругов образуют область G изолированную от других кругов, тогда G содержит точно n собственных значений матрицы A . Объединение всех кругов Гершгорина содержит все собственные значения матрицы A . Иногда возможно выделить круг, который содержит единственное собственное значение. Это возможно для матриц имеющих специальные свойства, а именно: если при некотором значении n для всех $k=1, \dots, N$ ($k \neq n$), справедливы неравенства

$$|a_{kk} - a_{nn}| < \text{sum}_k + \text{sum}_n, \quad (3.3)$$

то круг Гершгорина $|\lambda - a_{nn}| \leq \text{sum}_n$ содержит единственное собственное значение.

Пример 3.1 (локализация собственных значений)

Рассмотрим вещественную симметричную матрицу

$$A = \begin{pmatrix} 1 & 1 & 0 \\ 1 & 5 & 1 \\ 0 & 1 & 1 \end{pmatrix},$$

тогда $\text{sum}_1=1$, $\text{sum}_2=2$, $\text{sum}_3=1$. Согласно (3.2), собственные значения матрицы A лежат в интервалах $[0,2]$, $[3,7]$ и $[0,2]$. Условие (3.3) выполняется при $n=2$, следовательно, интервал $[3,7]$ содержит единственное собственное значение, а два других значения лежат в интервале $[0,2]$. Действительно, эта матрица имеет собственные значения 0.5505, 1, и 5.4495, с точностью до четырех знаков.

3.3 Степенной метод.

Степенной метод используется, в основном, для вычисления доминирующего собственного значения и соответствующего ему

собственного вектора. Он не является универсальным методом, но может быть полезен в ряде ситуаций, например, в случае больших разреженных матриц. В дополнение, схема этого метода демонстрирует некоторые важные аспекты вычисления собственных значений.

Степенной метод является итерационным процессом для нахождения собственных векторов матрицы **A**. Вначале, проведём некоторый предварительный анализ. Возьмём произвольный начальный вектор $\mathbf{x}^{(0)}$. Предположим, что все собственные значения различны по абсолютной величине. Разложим вектор $\mathbf{x}^{(0)}$ по собственным векторам матрицы **A**:

$$\mathbf{x}^{(0)} = \sum_{n=1}^N \alpha_n \mathbf{z}_n$$

Умножая вектор $\mathbf{x}^{(0)}$ k раз на матрицу **A**, получим

$$\begin{aligned} \mathbf{y}^{(k)} &= \mathbf{A}^k \mathbf{x}^{(0)} = \sum_{n=1}^N \alpha_n \mathbf{A}^k \mathbf{z}_n = \sum_{n=1}^N \alpha_n \lambda_n^k \mathbf{z}_n = \\ &= \lambda_N^k \left(\alpha_N \mathbf{z}_N + \sum_{n=1}^{N-1} \left(\frac{\lambda_n}{\lambda_N} \right)^k \alpha_n \mathbf{z}_n \right) \end{aligned}$$

Когда $k \rightarrow \infty$, $\mathbf{y}^{(k)}$ стремится к вектору коллинеарному с вектором $\mathbf{z}_N$. Поэтому, последовательные степени матрицы **A** дают информацию о её собственном значении λ_N . Величина этого собственного значения может быть легко найдена: учитывая второе свойство отношения Рэлея, $\rho(\mathbf{y}^{(k)})$ стремится к λ_N когда $k \rightarrow \infty$.

Основываясь на проведённом анализе, можно предложить следующую реализацию степенного метода:

- 1) задать начальный вектор $\mathbf{x}^{(0)}$
- 2) для каждого значения $k = 0, 1, \dots$ вычислить $\mathbf{y}^{(k+1)} = \mathbf{A} \mathbf{x}^{(k)}$
- 3) следующее приближение вычисляется как $\mathbf{x}^{(k+1)} = \mathbf{y}^{(k+1)} / \|\mathbf{y}^{(k+1)}\|$ (эта процедура называется нормализацией и она предназначена для того, чтобы предотвратить переполнение или исчезновение порядка)
- 4) если при некотором значении k , $\|\mathbf{x}^{(k+1)} - \mathbf{x}^{(k)}\| \leq \epsilon_p$, тогда $\mathbf{x}^{(k+1)}$ есть приближённый собственный вектор $\mathbf{z}_N$ и $s_N = \rho(\mathbf{x}^{(k+1)})$ есть приближённое максимальное по модулю собственное значение λ_N .

Сходимость степенного метода.

Если все $|\lambda_n|$ различны и $(\mathbf{z}_N, \mathbf{x}^{(0)}) \neq 0$, тогда

$$\lim_{k \rightarrow \infty} \mathbf{x}^{(k)} = \mathbf{z}_N$$

и справедлива следующая оценка

$$\frac{\| \mathbf{x}^{(k)} - \mathbf{z}_N \|}{\| \mathbf{x}^{(0)} - \mathbf{z}_N \|} \leq \left| \frac{\lambda_{N-1}}{\lambda_N} \right|^k = C^k$$

Таким образом, скорость сходимости степенного метода зависит от того насколько $|\lambda_{N-1}|$ меньше чем $|\lambda_N|$.

Пример 3.2 (степенной метод)

Рассмотрим следующую матрицу:

$$\mathbf{A} = \begin{pmatrix} 2 & -1 & 0 & 0 \\ -1 & 2 & -1 & 0 \\ 0 & -1 & 2 & -1 \\ 0 & 0 & -1 & 2 \end{pmatrix}$$

Выберем начальный вектор $\mathbf{x}^{(0)} = (1, 0, 0, 0)^T$ и $\varepsilon_p = 10^{-5}$. Результаты вычислений приведены в Таблице 3.1.

Таблица 3.1

k	$\ \mathbf{x}^{(k)} - \mathbf{x}^{(k-1)} \ $	$\rho(\mathbf{x}^{(k+1)})$
1	4.5950e-01	2.8000
2	2.6041e-01	3.1428
..		
18	1.8282e-03	3.6180
..		
35	7.4787e-06	3.6180

Видно, что приближения $\mathbf{x}^{(k)}$ довольно медленно сходятся к $\mathbf{z}_N$, так как эта матрица имеет собственные значения

$$\lambda_1 \approx 0.3820$$

$$\lambda_2 \approx 1.3819$$

$$\lambda_3 \approx 2.6181$$

$$\lambda_4 \approx 3.6180$$

и, следовательно, асимптотическая константа сходимости $C = \lambda_{N-1} / \lambda_N \approx 0.7236$.

Если нам нужно вычислить только собственное значение симметричной матрицы $\mathbf{A}$, то можно выполнить гораздо меньшее число итераций. Пусть $\mathbf{x}^{(k)}$ есть некоторое приближение к $\mathbf{z}_n$, такое, что $\mathbf{z}_n = \mathbf{x}^{(k)} + \mathbf{e}^{(k)}$ и $\|\mathbf{e}^{(k)}\| = \delta$. Отношение Рэля для вектора $\mathbf{x}^{(k)}$ имеет вид

$$\rho(\mathbf{x}^{(k)}) = \lambda_n (1 - 4(\mathbf{z}_n, \mathbf{e}^{(k)})^2 + O(\delta^3)) = \lambda_n + O(\delta^2).$$

Это выражение показывает, что $\rho(\mathbf{x}^{(k)})$ сходится квадратично к λ_n , в то время как приближения $\mathbf{x}^{(k)}$ в степенном методе сходятся линейно к $\mathbf{z}_n$. Например, как видно из примера 3.2, нужно только 18 итераций для того, чтобы вычислить λ_n с точностью до четырех знаков.

3.4 Метод обратной итерации.

Метод обратной итерации по существу является степенным методом, применённым к соответствующей обратной матрице. Рассмотрим кратко теоретические основы этого метода. Умножая выражение (3.1) на матрицу $\mathbf{A}^{-1}$, задачу на собственное значение можно записать в другом виде:

$$\mathbf{A}^{-1} \mathbf{z}_n = \frac{1}{\lambda_n} \mathbf{z}_n, \det(\mathbf{A}) \neq 0.$$

Это значит, что собственный вектор матрицы $\mathbf{A}$ является собственным вектором матрицы $\mathbf{A}^{-1}$ и $\lambda_n(\mathbf{A}^{-1}) = 1 / \lambda_n(\mathbf{A})$. Выберем некоторый начальный вектор $\mathbf{x}^{(0)}$ и разложим его по собственным векторам матрицы $\mathbf{A}$:

$$\mathbf{x}^{(0)} = \sum_{n=1}^N \alpha_n \mathbf{z}_n$$

Умножая вектор $\mathbf{x}^{(0)}$ k раз на матрицу $\mathbf{A}^{-1}$, получим

$$\begin{aligned} \mathbf{y}^{(k)} &= (\mathbf{A}^{-1})^k \mathbf{x}^{(0)} = \sum_{n=1}^N \alpha_n (\mathbf{A}^{-1})^k \mathbf{z}_n = \\ &= \sum_{n=1}^N \alpha_n \left(\frac{1}{\lambda_n} \right)^k \mathbf{z}_n = \left(\frac{1}{\lambda_1} \right)^k \left(\alpha_1 \mathbf{z}_1 + \sum_{n=2}^N \left(\frac{\lambda_1}{\lambda_n} \right)^k \alpha_n \mathbf{z}_n \right) \end{aligned}$$

Когда $k \rightarrow \infty$, $\mathbf{y}^{(k)}$ стремится к вектору коллинеарному с вектором $\mathbf{z}_1$. Поэтому, последовательные степени матрицы $\mathbf{A}^{-1}$ дают информацию о наименьшем по модулю собственном значении λ_1 матрицы $\mathbf{A}$. На

практике, вычислять матрицу A^{-1} нецелесообразно, так как решение системы $Ay=x$ относительно вектора y требует меньше затрат чем вычисление $y=A^{-1}x$. Суммируя все эти замечания, можно предложить следующую реализацию метода обратной итерации:

- 1) задать начальный вектор $x^{(0)}$
- 2) для каждого значения $k = 0, 1, \dots$ решить систему линейных уравнений $Ay^{(k+1)} = x^{(k)}$
- 3) следующее приближение вычисляется как $x^{(k+1)} = y^{(k+1)} / \|y^{(k+1)}\|$
- 4) если при некотором значении k , $\|x^{(k+1)} - x^{(k)}\| \leq \varepsilon_p$, тогда $x^{(k+1)}$ есть приближённый собственный вектор z_1 и $s_1 = \rho(x^{(k+1)})$ есть приближённое минимальное по модулю собственное значение λ_1 .

Сходимость метода обратной итерации.

Если все $|\lambda_n|$ различны и $(z_N, x^{(0)}) \neq 0$, тогда

$$\lim_{k \rightarrow \infty} x^{(k)} = z_1$$

и справедлива следующая оценка

$$\frac{\|x^{(k)} - z_1\|}{\|x^{(0)} - z_1\|} \leq \left| \frac{\lambda_1}{\lambda_2} \right|^k = C^k$$

Таким образом, скорость сходимости метода обратной итерации зависит от того насколько $|\lambda_1|$ меньше чем $|\lambda_2|$.

Пример 3.3 (метод обратной итерации)

Рассмотрим матрицу из примера 3.2. Выберем начальный вектор $x^{(0)} = (1, 0, 0, 0)^T$ и $\varepsilon_p = 10^{-5}$. Результаты вычислений приведены в Таблице 3.2.

Таблица 3.2

k	$\ x^{(k)} - x^{(k-1)}\ $	$\rho(x^{(k)})$
1	7.3444e-01	0.6666
2	3.5190e-01	0.4000
..		
5	6.8620e-03	0.3820
..		
10	1.0994e-05	0.3820

Для этой матрицы, метод обратной итерации сходится довольно быстро, так как асимптотическая константа сходимости $C=\lambda_1/\lambda_2 \approx 0.2764$. Если нам нужно только собственное значение, то достаточно 5-ти итераций чтобы вычислить его с точностью до четырёх знаков.

Метод обратной итерации требует больше вычислительных затрат по сравнению со степенным методом, так как необходимо решать систему линейных уравнений на каждом шаге итерационного процесса. Однако, это количество операций может быть значительно уменьшено, если учесть тот факт, что матрица **A** остаётся постоянной для всех итераций. Поэтому, в начале вычислительной процедуры следует разложить матрицу **A** (смотри 2.6.1), тогда решение системы уравнений в пункте 2) сводится к решению двух систем с треугольными матрицами. При таком подходе, вычислительные затраты для решения K систем уравнений сравнимы с затратами для решения одной системы.

3.5 Итерации со сдвигом начала.

Часто на практике, итерации применяются не к матрице **A**, а к матрице $\mathbf{B}=\mathbf{A}-\sigma\mathbf{I}$ ($\sigma=\text{const} \neq 0$ - сдвиг начала). Такое преобразование практически не меняет исходную задачу (3.1), так как собственные вектора матрицы **A** совпадают с собственными векторами матрицы **B** и $\lambda_n(\mathbf{A})=\lambda_n(\mathbf{B})+\sigma$. В этом случае, степенной метод сходится к некоторому собственному вектору $\mathbf{z}_n$ соответствующему собственному значению

$$|\lambda_n(\mathbf{B})| = \max_n |\lambda_n(\mathbf{A}) - \sigma|$$

и асимптотическая константа сходимости определяется как

$$C = \frac{\max_{m \neq n} |\lambda_m(\mathbf{A}) - \sigma|}{\max_n |\lambda_n(\mathbf{A}) - \sigma|}.$$

Метод обратной итерации сходится к некоторому собственному вектору $\mathbf{z}_n$ соответствующему собственному значению

$$|\lambda_n(\mathbf{B})| = \min_n |\lambda_n(\mathbf{A}) - \sigma|$$

и асимптотическая константа сходимости определяется как

$$C = \frac{\min_n |\lambda_n(\mathbf{A}) - \sigma|}{\min_{m \neq n} |\lambda_m(\mathbf{A}) - \sigma|}$$

В каких ситуациях следует использовать сдвиг начала?

Ситуация 1.

Пусть матрица $\mathbf{A}$ имеет $|\lambda_{N-1}| = |\lambda_N|$ (или они очень близки). Если мы применим степенной метод к этой матрице, то не получим сходимости ни к вектору $\mathbf{z}_N$ ни к вектору $\mathbf{z}_{N-1}$ (или сходимость будет очень медленной). Когда $\sigma \neq 0$, все $\lambda_n(\mathbf{B})$ различны по модулю и сходимость появится.

Пример 3.4 (степенной метод со сдвигом начала)

Рассмотрим следующую матрицу

$$\mathbf{A} = \begin{pmatrix} 1 & 1 & 0 \\ 1 & -1 & 1 \\ 0 & 1 & 1 \end{pmatrix}$$

Эта матрица имеет собственные значения

$$\lambda_1 = 1$$

$$\lambda_2 = \sqrt{3} \approx 1.7320$$

$$\lambda_3 = -\sqrt{3} \approx -1.7320$$

Применим степенной метод к матрице $\mathbf{A}$, начиная с вектора $\mathbf{x}^{(0)} = (1, 0, 0)^T$. Результаты вычислений приведены в Таблице 3.3.

Таблица 3.3

k	$\ \mathbf{r}^{(k)}\ $	$\rho(\mathbf{x}^{(k)})$
1	1.2247	1
2	1.3416	1
3	1.3887	1
4	1.4055	1
5	1.4113	1

Хорошо видно, что сходимость отсутствует. Теперь применим степенной метод к матрице $\mathbf{A} - \sigma \mathbf{I}$ ($\sigma=2$) с тем же самым начальным вектором. Результаты вычислений приведены в Таблице 3.4.

Таблица 3.4

k	$\ r^{(k)}\ $	$\rho(x^{(k)}) + \sigma$
1	1.2247	-1
2	0.4178	-16666
3	0.1141	-1.7272
4	0.0306	-1.7317
5	0.0082	-1.7320

Видно, что сдвиг начала обеспечил сходимость итераций и асимптотическая константа сходимости

$$C = \frac{|\lambda_1(A) - \sigma|}{|\lambda_3(A) - \sigma|} = \frac{1}{2 + \sqrt{3}} \approx 0.2679.$$

Ситуация 2.

Предположим, что нам надо найти собственное значение матрицы A ближайшее к заданному числу β , то есть такое $\lambda_m(A)$, что

$$|\lambda_m(A) - \beta| < \min_{n \neq m} |\lambda_n(A) - \beta|.$$

Тогда метод обратной итерации со сдвигом $\sigma = \beta$ сходится к собственному вектору z_m

Пример 3.5 (метод обратной итерации со сдвигом начала)

Рассмотрим снова матрицу из примера 3.2. Применим метод обратной итерации с начальным вектором $x^{(0)} = (1, 0, 0, 0)^T$ к матрице $A - \sigma I$ ($\sigma = 3.5$). Результаты вычислений приведены в Таблице 3.5.

Таблица 3.5

k	$\ x^{(k)} - x^{(k-1)}\ $	$\rho(x^{(k)}) + \sigma$
1	1.2895	3.5520
2	2.6116e-01	3.6171
3	3.3306e-02	3.6180
..		
6	7.8765e-05	3.6180

Мы выбрали сдвиг достаточно близко к собственному значению λ_4 , поэтому наблюдается быстрая сходимость к собственному вектору $\mathbf{z}_4$, так как асимптотическая константа сходимости

$$C = \frac{|\lambda_4(\mathbf{A}) - \sigma|}{|\lambda_3(\mathbf{A}) - \sigma|} \approx 0.104.$$

3.6 Применение ортогональных преобразований (QR метод).

QR метод является одним из наиболее часто используемых методов для вычисления собственных значений. Эта процедура имеет ряд полезных особенностей, а именно:

- 1) она позволяет вычислить все собственные значения матрицы
- 2) в случае одинаковых по модулю собственных значений, она всё же позволяет получить некоторую информацию о спектре матрицы
- 3) в случае симметричных матриц, вычисления сильно упрощаются

Основная идея QR метода выглядит просто. Сначала, для данной матрицы $\mathbf{A}$ вычисляется разложение $\mathbf{A} = \mathbf{A}^{(0)} = \mathbf{Q}\mathbf{R}$, где $\mathbf{Q}$ - ортогональная матрица и $\mathbf{R}$ - верхняя треугольная матрица (смотри 2.6.1). Затем вычисляется новая матрица $\mathbf{A}^{(1)} = \mathbf{R}\mathbf{Q}$. Далее процедура продолжается последовательно: для $\mathbf{A}^{(k)} = \mathbf{Q}^{(k)}\mathbf{R}^{(k)}$, мы вычисляем $\mathbf{A}^{(k+1)} = \mathbf{R}^{(k)}\mathbf{Q}^{(k)}$. Этот алгоритм выглядит довольно просто, но даёт очень интересный результат: если все $|\lambda_n|$ различны, то

$$\lim_{k \rightarrow \infty} \mathbf{A}^{(k)} = \mathbf{T},$$

где

$$\mathbf{T} = \begin{pmatrix} \lambda_N & * & . & . & * \\ & \lambda_{N-1} & & & . \\ & & . & & . \\ & 0 & & . & * \\ & & & & \lambda_1 \end{pmatrix} \quad (3.4)$$

Таким образом, мы можем вычислить все собственные значения в одном итерационном процессе. Однако, QR метод, в том виде, что мы рассмотрели, требует слишком больших вычислительных затрат. Если $\mathbf{A}$ - матрица общего вида, то каждое QR разложение потребует $O(N^3)$ операций. Для того чтобы сделать метод более эффективным, исходная матрица $\mathbf{A}$ сначала трансформируется в другую матрицу, для которой последующие QR разложения вычисляются за гораздо меньшее количество операций. Поэтому, QR метод начинается с преобразования исходной матрицы $\mathbf{A}$ в матрицу $\mathbf{H}_A$ в форме Хессенберга:

$$\mathbf{H}_A = \begin{pmatrix} * & . & . & . & * \\ * & . & . & . & . \\ & . & . & . & . \\ & 0 & . & . & . \\ & & & * & * \end{pmatrix}$$

После такого преобразования, QR разложение исходной матрицы $\mathbf{A}^{(0)} = \mathbf{H}_A$ можно вычислить за $O(N^2)$ операций. В дополнение, и это очень важно, все последующие матрицы $\mathbf{A}^{(k)}$ остаются в форме Хессенберга. Для симметричных матриц, форма Хессенберга является трёхдиагональной матрицей, что значительно уменьшает количество операций и позволяет эффективное хранение матриц. В качестве индикатора сходимости можно следить за уменьшением величины поддиагональных элементов $a_{n+1,n}$ в матрице $\mathbf{A}^{(k)}$ или можно следить за уменьшением величины

$$|a_{nn}^{(k+1)} - a_{nn}^{(k)}|, n=1, \dots, N.$$

Всё рассмотренное выше можно представить в виде следующего алгоритма:

- 1) представить матрицу $\mathbf{A}$ в форме Хессенберга: $\mathbf{A} \rightarrow \mathbf{H}_A = \mathbf{A}^{(0)}$
- 2) для каждого значения $k=0, 1, \dots$ вычислить QR разложение матрицы $\mathbf{A}^{(k)}$: $\mathbf{A}^{(k)} = \mathbf{Q}^{(k)} \mathbf{R}^{(k)}$ и следующее приближение $\mathbf{A}^{(k+1)} = \mathbf{R}^{(k)} \mathbf{Q}^{(k)}$
- 3) сформировать два вектора:

$$\mathbf{a}_1^{(k)} = (a_{21}^{(k)}, a_{32}^{(k)}, \dots, a_{N,N-1}^{(k)})^T$$

и

$$\mathbf{a}_2^{(k)} = (a_{11}^{(k)} - a_{11}^{(k-1)}, \dots, a_{NN}^{(k)} - a_{NN}^{(k-1)})^T.$$

Если

$$\| \mathbf{a}_1^{(k)} \| \leq \varepsilon_p \text{ или } \| \mathbf{a}_2^{(k)} \| \leq \varepsilon_p,$$

тогда элементы $\mathbf{a}_{n,n}$ матрицы $\mathbf{A}^{(k)}$ есть приближённые собственные значения матрицы $\mathbf{A}$ и

$$| \lambda_n(\mathbf{A}) - \mathbf{a}_{nn}^{(k)} | \leq \varepsilon_p$$

Сходимость QR метода.

Если все $|\lambda_n|$ различны, то матрицы $\mathbf{A}^{(k)}$ сходятся к верхней треугольной матрице $\mathbf{T}$ (смотри (3.4)), диагональные элементы которой являются собственными значениями матрицы $\mathbf{A}$. Для симметричных матриц, последовательность $\{ \mathbf{A}^{(k)} \}$ сходится к диагональной матрице. Скорость сходимости оценивается следующим образом:

$$\| \mathbf{T} - \mathbf{A}^{(k)} \| \leq \left(\max_{1 \leq n \leq N-1} \left| \frac{\lambda_n}{\lambda_{n+1}} \right| \right)^k$$

Легко видеть, что сходимость отсутствует, если хотя бы два собственных значения равны по модулю (или она очень медленная, если их абсолютные значения близки). Такая ситуация не такая уж редкая, особенно для несимметричных матриц, так как в этом случае собственные значения составляют комплексно-сопряжённые пары. Поэтому, в QR методе обычно применяется сдвиг начала, что часто обеспечивает сходимость.

Пример 3.6 (QR метод)

Рассмотрим следующую матрицу

$$\mathbf{A} = \begin{pmatrix} 1 & 1 & 0 & 0 \\ 1 & -1 & 1 & 0 \\ 0 & 0 & 2 & 1 \\ 0 & 0 & 1 & -2 \end{pmatrix}.$$

Собственные значения этой матрицы (с точностью до четырёх десятичных знаков):

$$\lambda_1 \approx 1.4142$$

$$\lambda_2 \approx -1.4142$$

$$\lambda_3 \approx 2.2360$$

$$\lambda_4 \approx -2.2360$$

Результат применения QR метода к матрице $\mathbf{A}-\sigma\mathbf{I}$ ($\sigma=1$) показан в Таблице 3.6.

Таблица 3.6

k	$\mathbf{a}_{nn}^{(k)} + \sigma$ $n=1, \dots, 4$	$\ \mathbf{a}_1^{(k)}\ _2$	$\ \mathbf{a}_2^{(k)}\ _2$
1	-1.0000 1.0000 1.0000 -1.0000	2.2360e+00	3.1622e+00
.			
3	2.0000 -2.0000 1.4138 -1.4138	1.0006e+00	1.4143e+00
.			
5	2.2307 -2.2307 1.4142 -1.4142	1.5384e-01	4.3514e-02
.			
10	2.2360 -2.2360 1.4142 -1.4142	1.2523e-03	2.9034e-06

3.7. Заключительные замечания.

В течении последних десятилетий был достигнут значительный прогресс в области вычислительной линейной алгебры. Это привело к созданию эффективного и надёжного программного обеспечения для

решения задач на собственное значение. Все основные пакеты программ, такие как IMSL[®], NAG[®] и LAPACK, включают в себя большой набор процедур для решения этой проблемы. Эти процедуры реализуют различные виды ортогональных разложений (одно из них, QR разложение, мы обсуждали) для различных типов матриц. В случае стандартной задачи на собственное значение (3.1) имеются вычислительные процедуры для вещественных симметричных и несимметричных матриц, комплексных общего вида и Эрмитовых матриц. В случае обобщённой задачи на собственное значение $\mathbf{A}\mathbf{z}_n = \lambda_n \mathbf{M}\mathbf{z}_n$ имеются вычислительные процедуры для следующих ситуаций:

- 1) матрицы $\mathbf{A}$ и $\mathbf{M}$ симметричные, и матрица $\mathbf{M}$ также положительно определённая
- 2) матрицы $\mathbf{A}$ и $\mathbf{M}$ вещественные или комплексные матрицы общего вида

ГЛАВА 4

Решение систем нелинейных уравнений

Список обозначений

$\mathbf{x} = (x_1, \dots, x_N)^T$	вектор (жирные строчные буквы)
N	число уравнений
$\ \cdot\ $	норма вектора или матрицы
ε_p	требуемая точность вычисления приближённого решения
$\mathbf{x}^* = (x_1^*, \dots, x_N^*)^T$	точное решение системы нелинейных уравнений ($\mathbf{f}(\mathbf{x}^*) \equiv \mathbf{0}$)
$\mathbf{x}^{(k)} = (x_1^{(k)}, \dots, x_N^{(k)})^T$	k-ое приближение к точному решению
$\mathbf{e}^{(k)} = \mathbf{x}^* - \mathbf{x}^{(k)} = (e_1^{(k)}, \dots, e_N^{(k)})^T$	ошибка k-го приближения
$\lambda_n(\mathbf{B})$	n-ое собственное значение матрицы $\mathbf{B}$
$s(\mathbf{B}) = \max_n \lambda_n(\mathbf{B}) $	спектральный радиус матрицы $\mathbf{B}$
$\mathbf{F}(\mathbf{x}) = \left\{ \frac{\partial f_n}{\partial x_m}(\mathbf{x}) \right\}$	матрица Якоби

Тема, обсуждаемая в этой главе, часто представляет довольно трудную проблему. Решение системы из N нелинейных уравнений часто требует гораздо больших усилий, чем просто решение N одномерных уравнений. Например, как мы видели в Главе 1, первая производная функции играет важную роль в создании итерационных алгоритмов. В случае N уравнений она заменяется матрицей из N^2 первых частных производных. Некоторые методы, применяемые для решения уравнения $\mathbf{f}(\mathbf{x})=0$, могут быть обобщены на случай систем нелинейных уравнений, однако простые методы, типа метода деления отрезка пополам, не имеют аналогов в многомерном случае. Поэтому, прежде чем использовать ту или иную итерационную схему, часто необходим некоторый

предварительный анализ, чтобы определить приближённое расположение решения.

Система нелинейных уравнений, в общем виде, записывается как

$$\mathbf{f}(\mathbf{x}) = \mathbf{0} \quad (4.1)$$

или в развёрнутой форме

$$f_1(x_1, \dots, x_N) = 0$$

$$\dots$$

$$f_N(x_1, \dots, x_N) = 0$$

В случае N уравнений, мы должны заменить интервалы на некоторые множества в $\mathbb{R}^N$, а именно:

$$S_c(\mathbf{x}_0, a) = \{\mathbf{x} \in \mathbb{R}^N : \|\mathbf{x} - \mathbf{x}_0\| \leq a\} \text{ (закрытый шар)}$$

$$S_o(\mathbf{x}_0, a) = \{\mathbf{x} \in \mathbb{R}^N : \|\mathbf{x} - \mathbf{x}_0\| < a\} \text{ (открытый шар)}$$

Форма шара зависит векторной нормы. Например, на рисунке 4.1 показаны различные формы шаров для $N=2$.

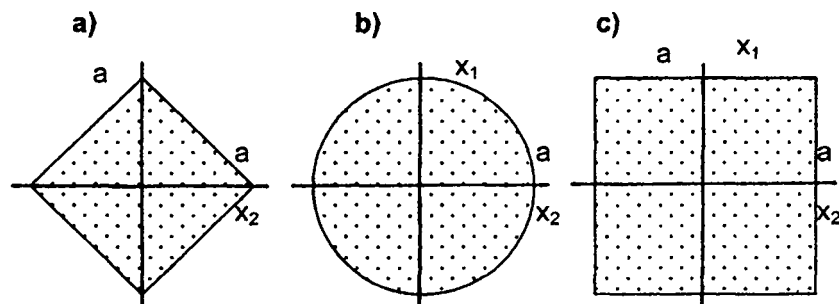


Рисунок 4.1. $S_c((0, 0), a)$ для некоторых векторных норм:

a) $\|\cdot\|_1$; b) $\|\cdot\|_2$; c) $\|\cdot\|_\infty$

4.1 Метод простой итерации.

По аналогии с (1.1), систему (4.1) можно представить в следующей эквивалентной форме:

$$\mathbf{x} = \mathbf{g}(\mathbf{x}) \quad (4.2)$$

или

$$\begin{aligned} x_1 &= g_1(x_1, \dots, x_N) \\ &\dots \dots \dots \\ x_N &= g_N(x_1, \dots, x_N), \end{aligned}$$

где $\mathbf{g}(\mathbf{x})$ - итерационная вектор функция векторного аргумента. Системы нелинейных уравнений часто возникают непосредственно в виде (4.2) (например, в численных схемах для дифференциальных уравнений), в этом случае никаких дополнительных усилий для преобразования уравнений (4.1) в систему (4.2) не требуется. Если продолжить аналогию с методом простой итерации для одного уравнения, то итерационный процесс, основанный на уравнении (4.2), можно организовать следующим образом:

- 1) выбирается некоторый начальный вектор $\mathbf{x}^{(0)} \in S_0(\mathbf{x}_0, a)$ (предполагается, что $\mathbf{x} \in S_0(\mathbf{x}_0, a)$)
- 2) последующие приближения вычисляются по формуле:

$$\mathbf{x}^{(k+1)} = \mathbf{g}(\mathbf{x}^{(k)}) \quad (4.3)$$

или

$$\begin{aligned} x_1^{(k+1)} &= g_1(x_1^{(k)}, \dots, x_N^{(k)}) \\ &\dots \dots \dots \\ x_N^{(k+1)} &= g_N(x_1^{(k)}, \dots, x_N^{(k)}), \\ k &= 0, 1, \dots \end{aligned}$$

- 3) если

$$\frac{\|\mathbf{x}^{(k+1)} - \mathbf{x}^{(k)}\|}{\|\mathbf{x}^{(k+1)}\|} \leq \varepsilon_p$$

или

$$\|\mathbf{x}^{(k+1)} - \mathbf{g}(\mathbf{x}^{(k)})\| \leq \varepsilon_p,$$

то итерационный процесс завершён и

$$\frac{\|\mathbf{x}^* - \mathbf{x}^{(k)}\|}{\|\mathbf{x}^*\|} \leq \varepsilon_p$$

Как и раньше, нам необходимо выяснить при каких условиях

$$\lim_{k \rightarrow \infty} \mathbf{x}^{(k)} = \mathbf{x}^*.$$

Обсудим этот вопрос, выполнив простой анализ. Вначале мы введём ошибку k -го приближения как $\mathbf{e}^{(k)} = \mathbf{x}^{(k)} - \mathbf{x}^*$. Тогда мы можем записать

$$\begin{aligned} \mathbf{x}^{(k)} &= \mathbf{x}^* + \mathbf{e}^{(k)}, \\ (x_1^{(k)}, \dots, x_N^{(k)}) &= (x_1^* + e_1^{(k)}, \dots, x_N^* + e_N^{(k)}) \end{aligned}$$

и

$$\begin{aligned} \mathbf{x}^{(k+1)} &= \mathbf{x}^* + \mathbf{e}^{(k+1)} \\ (x_1^{(k+1)}, \dots, x_N^{(k+1)}) &= (x_1^* + e_1^{(k+1)}, \dots, x_N^* + e_N^{(k+1)}) \end{aligned}$$

Подставим эти выражения в (4.3) и разложим $\mathbf{g}(\mathbf{x}^* + \mathbf{e}^{(k)})$ по степеням $\mathbf{e}^{(k)}$ в окрестности $\mathbf{x}^*$ как функцию векторного аргумента (предполагая, что все частные производные функции $\mathbf{g}(\mathbf{x})$ непрерывны). Учитывая также, что $\mathbf{x}^* = \mathbf{g}(\mathbf{x}^*)$ мы получим

$$\mathbf{e}_n^{(k+1)} = \sum_{m=1}^N \frac{\partial g_n}{\partial x_m}(\mathbf{x}^*) e_m^{(k)} + O((e_n^{(k)})^2)$$

или в матричной форме

$$\mathbf{e}^{(k+1)} = \mathbf{B} \mathbf{e}^{(k)} + O(\|\mathbf{e}^{(k)}\|^2), \quad (4.4)$$

где

$$\mathbf{B} = \{b_{nm}\} = \left\{ \frac{\partial g_n}{\partial x_m}(\mathbf{x}^*) \right\} - \text{итерационная матрица.}$$

Если норма ошибки $\|\mathbf{e}^{(k)}\|$ достаточно мала, то вторым слагаемым в правой части выражения (4.4) можно пренебречь и тогда оно совпадает с выражением (2.16). Следовательно, условие сходимости итерационного процесса (4.3) вблизи точного решения, описывается Теоремой 2.1.

Сходимость метода простой итерации.

Необходимое и достаточное условие для сходимости итерационного процесса (4.3):

$$s(\mathbf{B}) = \max_n |\lambda_n(\mathbf{B})| < 1$$

и достаточное условие:

$$\| \mathbf{B} \| < 1 \quad (4.5)$$

Эти условия имеют скорее теоретическое, чем практическое значение, так как мы не знаем $\mathbf{x}^*$. По аналогии с (1.11), получим условие, которое может быть полезным. Пусть $\mathbf{x} \in S_0(\mathbf{x}_0, \mathbf{a})$ и матрица Якоби для функции $\mathbf{g}(\mathbf{x})$

$$\mathbf{C}(\mathbf{x}) = \{c_{nm}(\mathbf{x})\} = \left\{ \frac{\partial g_n}{\partial x_m}(\mathbf{x}) \right\} = \begin{pmatrix} \frac{\partial g_1}{\partial x_1}(\mathbf{x}) & \frac{\partial g_1}{\partial x_2}(\mathbf{x}) & \dots & \frac{\partial g_1}{\partial x_N}(\mathbf{x}) \\ \frac{\partial g_2}{\partial x_1}(\mathbf{x}) & \frac{\partial g_2}{\partial x_2}(\mathbf{x}) & \dots & \frac{\partial g_2}{\partial x_N}(\mathbf{x}) \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial g_N}{\partial x_1}(\mathbf{x}) & \frac{\partial g_N}{\partial x_2}(\mathbf{x}) & \dots & \frac{\partial g_N}{\partial x_N}(\mathbf{x}) \end{pmatrix}$$

существует для всех $\mathbf{x} \in S_0(\mathbf{x}_0, \mathbf{a})$ (заметим, что $\mathbf{C}(\mathbf{x}^*) = \mathbf{B}$). Если элементы матрицы $\mathbf{C}(\mathbf{x})$ удовлетворяют неравенству

$$|c_{nm}(\mathbf{x})| < \frac{1}{N}, \quad n, m = 1, \dots, N \quad (4.6)$$

для всех $\mathbf{x} \in S_0(\mathbf{x}_0, \mathbf{a})$, тогда достаточное условие (4.5) также выполняется для любой матричной нормы.

Пример 4.1 (метод простой итерации).

Рассмотрим следующую систему уравнений:

$$\begin{cases} f_1(x_1, x_2) = x_2(x_1 - 1) - 1 = 0 \\ f_2(x_1, x_2) = x_1^2 - x_2^2 - 1 = 0 \end{cases} \quad (4.7)$$

Одна из возможностей представить эту систему в эквивалентной форме (4.2) - выразить x_1 из первого уравнения и x_2 из второго уравнения:

$$\begin{cases} x_1 = 1 + 1/x_2 = g_1(x_1, x_2) \\ x_2 = \sqrt{x_1^2 - 1} = g_2(x_1, x_2) \end{cases}$$

Тогда итерационная схема имеет вид

$$\begin{aligned} x_1^{(k+1)} &= 1 + 1/x_2^{(k)} \\ x_2^{(k+1)} &= \sqrt{(x_1^{(k)})^2 - 1}, \quad k = 0, 1, \dots \end{aligned}$$

Точное решение $\mathbf{x}^* \in S_o((2, 2), 1)$. Выберем начальный вектор $\mathbf{x}^{(0)} = (2, 2)$ и $\varepsilon_p = 10^{-5}$. Результаты вычислений представлены в Таблице 4.1.

Таблице 4.1

k	$\mathbf{x}^{(k)}$	$\ \mathbf{x}^{(k)} - \mathbf{x}^{(k-1)}\ _2 / \ \mathbf{x}^{(k)}\ _2$	$\ \mathbf{f}(\mathbf{x}^{(k)})\ _2$
1	1.50000 1.73205	2.0056e-01	1.7551e-00
10	1.69258 1.34646	4.4561e-02	8.5105e-02
20	1.71914 1.40036	4.4992e-03	9.0029e-03
30	1.71642 1.39483	4.5324e-04	9.0258e-04
40	1.71669 1.39536	4.5665e-05	9.0982e-05
47	1.71667 1.39532	6.2770e-06	4.4503e-05

Эти результаты показывают, что сходимость довольно медленная. Для того чтобы получить количественную характеристику сходимости,

Пример 4.2 (модифицированный метод простой итерации)

Модифицированная простая итерация для системы (4.7) представляется в виде

$$x_1^{(k+1)} = 1 + 1/x_2^{(k)}$$

$$x_2^{(k+1)} = \sqrt{(x_1^{(k+1)})^2 - 1}, \quad k = 0, 1, \dots$$

Как и прежде выберем начальный вектор $x^{(0)} = (2, 2)$ и $\varepsilon_p = 10^{-5}$. Результаты вычислений представлены в Таблице 4.2.

Таблица 4.2

k	$x^{(k)}$	$\ x^{(k)} - x^{(k-1)}\ _2 / \ x^{(k)}\ _2$	$\ f(x^{(k)})\ _2$
1	1.50000 1.11803	3.5844e-01	4.4098e-01
.			
10	1.72076 1.40036	7.5781e-03	9.3355e-03
.			
20	1.71671 1.39538	7.6678e-05	5.9607e-05
.			
25	1.71667 1.39533	7.7251e-06	9.5041e-06

Небольшое изменение порядка вычислений привело к уменьшению количества итераций в два раза, а значит и к уменьшению количества операций в два раза.

4.2 Метод Ньютона.

В Главе 1 мы рассматривали метод Ньютона, который является очень эффективным методом для решения скалярного уравнения. Этот метод обобщается и на случай системы уравнений. Пусть $x^* \in S_0(x_0, a)$ и

первые частные производные векторной функции $\mathbf{f}(\mathbf{x})$ существуют, тогда можно разложить $\mathbf{f}_n(\mathbf{x})$ в окрестности некоторого приближения $\mathbf{x}^{(k)} \in S_0(\mathbf{x}_0, \mathbf{a})$ по степеням $(\mathbf{x} - \mathbf{x}^{(k)})$:

$$\begin{aligned} f_n(x_1, \dots, x_N) &= f_n(x_1^{(k)}, \dots, x_N^{(k)}) + \\ &\sum_{m=1}^N \frac{\partial f_n}{\partial x_m}(\mathbf{x}^{(k)})(x_m - x_m^{(k)}) + O((x_n - x_n^{(k)})^2) = \\ &p_n(x_1, \dots, x_N) + O((x_n - x_n^{(k)})^2), \\ n &= 1, \dots, N \end{aligned}$$

Это разложение можно также представить в матричной форме:

$$\begin{aligned} \mathbf{f}(\mathbf{x}) &= \mathbf{f}(\mathbf{x}^{(k)}) + \mathbf{F}(\mathbf{x}^{(k)})(\mathbf{x} - \mathbf{x}^{(k)}) + O(\|\mathbf{x} - \mathbf{x}^{(k)}\|^2) = \\ &\mathbf{p}(\mathbf{x}) + O(\|\mathbf{x} - \mathbf{x}^{(k)}\|^2), \end{aligned}$$

где

$$\mathbf{F}(\mathbf{x}) = \left\{ \frac{\partial f_n}{\partial x_m}(\mathbf{x}) \right\} = \begin{pmatrix} \frac{\partial f_1}{\partial x_1}(\mathbf{x}) & \frac{\partial f_1}{\partial x_2}(\mathbf{x}) & \dots & \frac{\partial f_1}{\partial x_N}(\mathbf{x}) \\ \frac{\partial f_2}{\partial x_1}(\mathbf{x}) & \frac{\partial f_2}{\partial x_2}(\mathbf{x}) & \dots & \frac{\partial f_2}{\partial x_N}(\mathbf{x}) \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial f_N}{\partial x_1}(\mathbf{x}) & \frac{\partial f_N}{\partial x_2}(\mathbf{x}) & \dots & \frac{\partial f_N}{\partial x_N}(\mathbf{x}) \end{pmatrix}$$

есть матрица Якоби для функции $\mathbf{f}(\mathbf{x})$. Следующее приближение определяется из условия $\mathbf{p}(\mathbf{x}^{(k+1)}) = \mathbf{0}$ или

$$\mathbf{F}(\mathbf{x}^{(k)})(\mathbf{x}^{(k+1)} - \mathbf{x}^{(k)}) + \mathbf{f}(\mathbf{x}^{(k)}) = \mathbf{0}. \quad (4.9)$$

На практике, вычисления по методу Ньютона организуются следующим образом:

1) выбирается некоторый начальный вектор $\mathbf{x}^{(0)}$.

- 2) для каждого значения $k = 0, 1, \dots$ решается система линейных уравнений

$$\mathbf{F}(\mathbf{x}^{(k)}) \Delta \mathbf{x}^{(k+1)} = -\mathbf{f}(\mathbf{x}^{(k)}) \quad (4.10)$$

относительно вектора $\Delta \mathbf{x}^{(k+1)}$, где $\Delta \mathbf{x}^{(k+1)} = \mathbf{x}^{(k+1)} - \mathbf{x}^{(k)}$. Следующее приближение вычисляется очень легко: $\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + \Delta \mathbf{x}^{(k+1)}$.

- 3) если

$$\frac{\|\mathbf{x}^{(k+1)} - \mathbf{x}^{(k)}\|}{\|\mathbf{x}^{(k+1)}\|} \leq \varepsilon_p$$

или

$$\|\mathbf{f}(\mathbf{x}^{(k+1)})\| \leq \varepsilon_p,$$

то итерационный процесс завершён и

$$\frac{\|\mathbf{x}^* - \mathbf{x}^{(k)}\|}{\|\mathbf{x}^*\|} \leq \varepsilon_p$$

Для решения системы линейных уравнений с матрицей $\mathbf{F}(\mathbf{x}^{(k)})$ следует использовать прямые методы, так как эта матрица изменяется с k и маловероятно, чтобы какой-нибудь итерационный метод сходиллся при каждом значении k . При этом тип разложения зависит свойств и структуры матрицы $\mathbf{F}(\mathbf{x}^{(k)})$. Теперь рассмотрим вопрос сходимости метода Ньютона.

Теорема о сходимости метода Ньютона (Канторович).

Пусть функция $\mathbf{f}(\mathbf{x})$ дифференцируема в некоторой области $S_0(\mathbf{x}_0, a)$ с матрицей Якоби $\mathbf{F}(\mathbf{x})$. Предположим также, что матрица $\mathbf{F}(\mathbf{x})$ обратима для всех $\mathbf{x} \in S_0(\mathbf{x}_0, a)$ и существуют вещественные положительные числа γ_1, γ_2 , и γ_3 такие, что

$$\|\mathbf{F}^{-1}(\mathbf{x}^{(0)})\| \leq \gamma_1$$

$$\|\mathbf{F}^{-1}(\mathbf{x}^{(0)})\mathbf{f}(\mathbf{x}^{(0)})\| \leq \gamma_2$$

и

$$\gamma_3 \leq \frac{1}{2\gamma_1\gamma_2}$$

Пусть также функция $\mathbf{f}(\mathbf{x})$ дважды дифференцируема и выполняется условие

$$\|D(\mathbf{x})\| = \left\| \frac{\partial^2 f_n}{\partial x_m \partial x_l}(\mathbf{x}) \right\| \leq \gamma_3$$

$$n = 1, \dots, N$$

для всех $\mathbf{x} \in S_0(\mathbf{x}^{(0)}, a)$, где

$$b = \frac{1}{\gamma_1 \gamma_3} \left(1 - \sqrt{1 - 2\gamma_1 \gamma_2 \gamma_3} \right).$$

Если шар $S_0(\mathbf{x}^{(0)}, b) \subset S_0(\mathbf{x}_0, a)$, тогда последовательность $\{\mathbf{x}^{(k)}\}$, генерируемая методом Ньютона с начальным вектором $\mathbf{x}^{(0)}$, сходится к $\mathbf{x} \in S_0(\mathbf{x}^{(0)}, b)$ и существует некоторая константа $C > 0$ такая, что

$$\|\mathbf{x}^* - \mathbf{x}^{(k+1)}\| \leq C \|\mathbf{x}^* - \mathbf{x}^{(k)}\|^2.$$

Это говорит о квадратичной сходимости, также как и в случае одного уравнения

Пример 4.3 (метод Ньютона)

Рассмотрим систему уравнений (4.7). Матрица Якоби для этой системы имеет простой вид:

$$F(\mathbf{x}) = \begin{pmatrix} x_2 & x_1 - 1 \\ 2x_1 & -2x_2 \end{pmatrix}$$

Эта матрица обратима для $\mathbf{x} \in \{\mathbf{x} : x_1 > 1, x_2 > 0\} = S_0(\mathbf{x}_0, a)$ и

$$F^{-1}(\mathbf{x}) = \frac{1}{x_2^2 + x_1(x_1 - 1)} \begin{pmatrix} x_2 & 0.5(x_1 - 1) \\ x_1 & -0.5x_2 \end{pmatrix}$$

Выберем начальное приближение $\mathbf{x}^{(0)} = (2, 2)$, тогда

$$\|F^{-1}(\mathbf{x}^{(0)})\|_{\infty} = 0.5 \text{ и } \|F^{-1}(\mathbf{x}^{(0)})f(\mathbf{x}^{(0)})\|_{\infty} = 0.5.$$

Числа γ_1 , γ_2 , и γ_3 определяются как $\gamma_1 = 0.5$, $\gamma_2 = 0.5$, и $\gamma_3 = 2$. Матрицы $D_n(\mathbf{x})$ для данной системы имеют следующий вид

$$D_1(x) = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \text{ и } D_2(x) = \begin{pmatrix} 2 & 0 \\ 0 & -2 \end{pmatrix},$$

таким образом, условие $\|D_n(x)\| \leq \gamma_3$ выполняется и $b=1$. Шар $S_0(x^{(0)}, 1) \in S_0(x_0, a)$, поэтому приближения, генерируемые по методу Ньютона, сходятся к точному решению квадратично, начиная с выбранного начального приближения.

Итерационная схема для данной системы уравнений записывается в виде

$$\begin{pmatrix} x_2^{(k)} & x_1^{(k)} - 1 \\ 2x_1^{(k)} & -2x_2^{(k)} \end{pmatrix} \begin{pmatrix} \Delta x_1^{(k+1)} \\ \Delta x_2^{(k+1)} \end{pmatrix} = \begin{pmatrix} -x_2^{(k)}(x_1^{(k)} - 1) + 1 \\ 1 - (x_1^{(k)})^2 + (x_2^{(k)})^2 \end{pmatrix},$$

$$\begin{pmatrix} x_1^{(k+1)} \\ x_2^{(k+1)} \end{pmatrix} = \begin{pmatrix} x_1^{(k)} + \Delta x_1^{(k+1)} \\ x_2^{(k)} + \Delta x_2^{(k+1)} \end{pmatrix},$$

$$x^{(0)} = \begin{pmatrix} 2 \\ 2 \end{pmatrix}, k = 0, 1, \dots$$

Результаты вычислений представлены в Таблице 4.3.

Таблица 4.3

k	$x^{(k)}$	$\ x^{(k)} - x^{(k-1)}\ _2 / \ x^{(k)}\ _2$	$\ f(x^{(k)})\ _2$
1	1.75000 1.50000	2.4253e-01	2.2534e-01
2	1.71710 1.39912	4.7904e-02	9.6806e-03
3	1.71667 1.39534	1.7207e-03	1.4209e-05
4	1.71667 1.39533	1.8439e-06	1.5512e-011

Этот пример показывает, что когда начальное приближение выбрано достаточно близко к x , метод Ньютона обеспечивает быструю сходимость.

4.3 Метод с кубической сходимостью.

Один из путей построения метода с кубической сходимостью это провести аналогию с методами третьего порядка для одномерного уравнения. При этом имеет смысл рассматривать схемы, использующие только первую производную функции, так как в случае N уравнений вторая производная скалярной функции заменяется матрицей содержащей N^2 частных производных второго порядка, что приводит к очень громоздким алгоритмам. Поэтому многоточечная итерация (1.25) является наиболее приемлемым выбором. Перепишем эту схему в следующем виде

$$\begin{aligned}f'(x_k)(y_k - x_k) + f(x_k) &= 0 \\f'(x_k)(x_{k+1} - y_k) + f(y_k) &= 0\end{aligned}$$

В многомерном случае мы работаем с векторами из R^N и естественным аналогом $f'(x)$ является матрица Якоби $F(x)$, тогда предыдущее выражение превращается в

$$\begin{aligned}F(x^{(k)})(y^{(k)} - x^{(k)}) + f(x^{(k)}) &= 0 \\F(x^{(k)})(x^{(k+1)} - y^{(k)}) + f(y^{(k)}) &= 0\end{aligned}$$

Следует заметить, что метод Ньютона для системы уравнений можно построить, поступая аналогичным образом (смотри (4.9)). Теперь, вместо системы (4.10) мы решаем две линейные системы для каждого значения k :

$$\begin{aligned}F(x^{(k)})\Delta y^{(k)} &= -f(x^{(k)}) \\y^{(k)} &= x^{(k)} + \Delta y^{(k)} \\F(x^{(k)})\Delta x^{(k+1)} &= -f(y^{(k)}) \\x^{(k+1)} &= y^{(k)} + \Delta x^{(k+1)}\end{aligned}\tag{4.11}$$

Матрица $F(x^{(k)})$ одинакова для обеих систем, поэтому предварительно можно разложить её на множители и затем использовать эти множители для решения двух систем. Такой подход приводит к тому, что при

больших значениях N вычислительные затраты на решение второй системы определяются в основном затратами на вычисление $f(y^{(k)})$. Если этот метод сходится, то имеет место сходимость третьего порядка, то есть

$$\|x^* - x^{(k+1)}\| \leq C \|x^* - x^{(k)}\|^3$$

Пример 4.4 (кубическая сходимость)

Рассмотрим систему уравнений (4.7). Матрица Якоби для этой системы приведена в примере 4.3 и итерационная схема (4.11) имеет вид

$$\begin{pmatrix} x_2^{(k)} & x_1^{(k)} - 1 \\ 2x_1^{(k)} & -2x_2^{(k)} \end{pmatrix} \begin{pmatrix} \Delta y_1^{(k)} \\ \Delta y_2^{(k)} \end{pmatrix} = \begin{pmatrix} -x_2^{(k)}(x_1^{(k)} - 1) + 1 \\ 1 - (x_1^{(k)})^2 + (x_2^{(k)})^2 \end{pmatrix},$$

$$\begin{pmatrix} y_1^{(k)} \\ y_2^{(k)} \end{pmatrix} = \begin{pmatrix} x_1^{(k)} + \Delta y_1^{(k)} \\ x_2^{(k)} + \Delta y_2^{(k)} \end{pmatrix},$$

$$\begin{pmatrix} x_2^{(k)} & x_1^{(k)} - 1 \\ 2x_1^{(k)} & -2x_2^{(k)} \end{pmatrix} \begin{pmatrix} \Delta x_1^{(k+1)} \\ \Delta x_2^{(k+1)} \end{pmatrix} = \begin{pmatrix} -y_2^{(k)}(y_1^{(k)} - 1) + 1 \\ 1 - (y_1^{(k)})^2 + (y_2^{(k)})^2 \end{pmatrix},$$

$$\begin{pmatrix} x_1^{(k+1)} \\ x_2^{(k+1)} \end{pmatrix} = \begin{pmatrix} y_1^{(k)} + \Delta x_1^{(k+1)} \\ y_2^{(k)} + \Delta x_2^{(k+1)} \end{pmatrix}, \quad x^{(0)} = \begin{pmatrix} 2 \\ 2 \end{pmatrix},$$

$$k = 0, 1, \dots$$

Результаты вычислений представлены в Таблице 4.4.

Таблица 4.4

k	$x^{(k)}$	$\ x^{(k)} - x^{(k-1)}\ _2 / \ x^{(k)}\ _2$	$\ f(x^{(k)})\ _2$
1	1.72395 1.42708	3.4596e-02	7.2550e-02
2	1.71667 1.39534	1.4605e-04	2.0578e-05
3	1.71667 1.39533	3.2996e-12	2.2204e-16

Этот метод обеспечивает ещё более быструю сходимость по сравнению с методом Ньютона, но для данной системы он менее эффективен, так как требует больше вычислений на итерацию. Метод с кубической сходимостью может иметь предпочтение перед методом Ньютона, когда число уравнений становится относительно большим.

4.4 Модификации метода Ньютона.

Часто на практике выражения для элементов матрицы Якоби имеют сложный вид и их вычисление требует гораздо больших затрат по сравнению с вычислением функции $f(\mathbf{x})$. По аналогии с одномерным случаем, можно использовать разностные отношения для аппроксимации частных производных, которые образуют матрицу Якоби. Применяя этот метод, мы заменяем матрицу $\mathbf{F}(\mathbf{x})$ приближённой матрицей $\mathbf{D}(\mathbf{x})$, которая имеет следующий вид:

$$\mathbf{D}(\mathbf{x}^{(k)}) = \{d_{nm}(\mathbf{x}^{(k)})\} = \left\{ \frac{f_n(\mathbf{x}_1^{(k)}, \dots, \mathbf{x}_m^{(k)} + h, \dots, \mathbf{x}_N^{(k)}) - f_n(\mathbf{x}_1^{(k)}, \dots, \mathbf{x}_N^{(k)})}{h} \right\}, \quad (4.12)$$

где h - малый параметр. Если вместо матрицы $\mathbf{F}(\mathbf{x}^{(k)})$ в (4.10) подставить матрицу $\mathbf{D}(\mathbf{x}^{(k)})$, определённую в (4.12), мы получим конечно-разностный метод Ньютона. Этот метод обеспечивает, по крайней мере, линейную сходимость, если значение параметра h достаточно мало. Для вычисления элементов матрицы $\mathbf{D}(\mathbf{x}^{(k)})$ можно применить процедуру аналогичную той, что использовалась в методе секущих (1.26). Однако, применяя такой подход мы можем столкнуться с ситуацией деления очень малых чисел, когда приближенное решение близко к $\mathbf{x}$ и это может привести к значительному возмущению матрицы $\mathbf{D}(\mathbf{x}^{(k)})$. В результате этого, сходимость итерационного процесса может замедлиться. Метод с кубической сходимостью можно также модифицировать, заменяя матрицу $\mathbf{F}(\mathbf{x}^{(k)})$ в (4.11) на матрицу $\mathbf{D}(\mathbf{x}^{(k)})$.

Шаг (2) в методе Ньютона предусматривает решение системы линейных уравнений на каждой итерации. Когда N достаточно большое это может потребовать значительных вычислительных затрат. Поэтому в некоторых ситуациях имеет смысл несколько упростить метод

Ньютона и тем самым уменьшить количество вычислений необходимое для решения линейных систем с матрицей Якоби. Один из простейших способов такой модификации – вычислить матрицу $F(x^{(0)})$ и затем использовать её вместо матрицы $F(x^{(k)})$ в последующих итерациях. Тогда мы можем разложить матрицу $F(x^{(0)})$ на множители один раз и использовать эти множители для решения систем уравнений на последующих итерациях. Другой несколько более сложный способ модифицировать метод Ньютона состоит в том, чтобы изменять матрицу $F(x^{(k)})$ не на каждой итерации, а через m итераций. Это значит, что вместо матрицы $F(x^{(k)})$ используется матрица

$$F_m(x^{(k)}) = \begin{cases} F(x^{(k)}), & \text{if } k = pm \\ F(x^{(p)}), & \text{if } k \neq pm \end{cases},$$

$$p = \left[\frac{k}{m} \right],$$

где $[y]$ обозначает целую часть числа y .

Пример 4.5 (модифицированный метод Ньютона)

Рассмотрим систему уравнений (4.7). Матрица Якоби для этой системы приведена в примере 4.3. Если мы выберем начальное приближение $x^{(0)} = (2, 2)$, тогда

$$F(x^{(0)}) = \begin{pmatrix} 2 & 1 \\ 4 & -4 \end{pmatrix}$$

и итерационная схема упрощённого метода Ньютона имеет вид

$$\begin{pmatrix} 2 & 1 \\ 4 & -4 \end{pmatrix} \begin{pmatrix} \Delta x_1^{(k+1)} \\ \Delta x_2^{(k+1)} \end{pmatrix} = \begin{pmatrix} -x_2^{(k)}(x_1^{(k)} - 1) + 1 \\ 1 - (x_1^{(k)})^2 + (x_2^{(k)})^2 \end{pmatrix}, \quad (4.13)$$

$$\begin{pmatrix} x_1^{(k+1)} \\ x_2^{(k+1)} \end{pmatrix} = \begin{pmatrix} x_1^{(k)} + \Delta x_1^{(k+1)} \\ x_2^{(k)} + \Delta x_2^{(k+1)} \end{pmatrix}, \quad k = 0, 1, \dots$$

Результаты вычислений представлены в Таблице 4.5.

Таблица 4.5

k	$\mathbf{x}^{(k)}$	$\ \mathbf{x}^{(k)} - \mathbf{x}^{(k-1)}\ _2 / \ \mathbf{x}^{(k)}\ _2$	$\ \mathbf{f}(\mathbf{x}^{(k)})\ _2$
1	1.75000 1.50000	2.4253e-01	2.2534e-01
2	1.72396 1.42708	7.4598e-02	7.2550e-02
3	1.71829 1.40528	1.0150e-02	2.4194e-02
...	...	...	...
9	1.71667 1.39534	8.1816e-06	2.2523e-05

Результаты вычислений указывают на линейную сходимость. На самом деле эта итерационная схема представляет собой простую итерацию (4.3), так как мы можем переписать выражение (4.13) в следующей форме

$$\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} - \mathbf{F}^{-1}(\mathbf{x}^{(0)})\mathbf{f}(\mathbf{x}^{(k)}) = \mathbf{g}(\mathbf{x}^{(k)})$$

или

$$x_1^{(k+1)} = x_1^{(k)} - \frac{1}{3}f_1(x_1^{(k)}, x_2^{(k)}) - \frac{1}{12}f_2(x_1^{(k)}, x_2^{(k)}) = g_1(x_1^{(k)}, x_2^{(k)})$$

$$x_2^{(k+1)} = x_2^{(k)} - \frac{1}{3}f_1(x_1^{(k)}, x_2^{(k)}) + \frac{1}{6}f_2(x_1^{(k)}, x_2^{(k)}) = g_2(x_1^{(k)}, x_2^{(k)})$$

Проводя такой же анализ как в примере 4.1, получим оценку итерационной матрицы $\mathbf{B}$ определённой в (4.4):

$$\mathbf{B} = \mathbf{C}(\mathbf{x}^*) \approx \mathbf{C}(\mathbf{x}^{(9)}) \approx \begin{pmatrix} 0.25 & -0.006 \\ 0.11 & 0.30 \end{pmatrix}.$$

Это указывает на более быструю сходимость по сравнению с методом простой итерации из примера 4.1, так как теперь $s(\mathbf{B}) \approx 0.3$.

4.5 Повышение надёжности метода Ньютона.

Преимуществом метода Ньютона является то, что он обеспечивает быструю сходимость. Однако квадратичная сходимость имеет место

только тогда, когда начальное приближение выбрано достаточно близко к точному решению. Существует довольно простой способ расширить набор начальных приближений, которые порождают итерационные последовательности $\{\mathbf{x}^{(k)}\}$, быстро сходящиеся к точному решению. Идея этого метода основана на том факте, что система уравнений (4.1) имеет решение $\mathbf{x}$, когда функция определённая как

$$w(\mathbf{x}) = w(x_1, \dots, x_N) = \sum_{n=1}^N f_n^2(x_1, \dots, x_N),$$

имеет минимальное значение равное нулю (рисунок 4.2). Для того чтобы найти этот минимум рассмотрим приращение $\Delta \mathbf{x}^{(k+1)}$, генерируемое методом Ньютона, как направление для поиска минимума функции $w(\mathbf{x})$. Если $w(\mathbf{x}^{(k)} + \Delta \mathbf{x}^{(k+1)}) \geq w(\mathbf{x}^{(k)})$, тогда мы будем делить приращение пополам до тех пор, пока не достигнем уменьшения значения функции $w(\mathbf{x})$. Теперь шаг 2) метода Ньютона следует немного изменить. После решения системы (4.10) будем вычислять вектор

$$\mathbf{d}^{(m)} = \frac{1}{2^m} \Delta \mathbf{x}^{(k+1)}, m = 0, 1, \dots$$

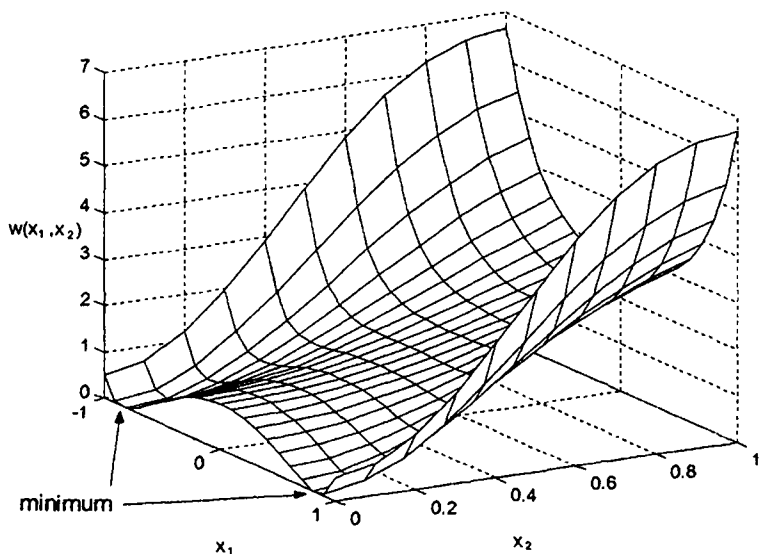


Рисунок 4.2. Пример функции $w(\mathbf{x})$ для двумерной проблемы.

до тех пор, пока значение $w(\mathbf{x}^{(k)} + \mathbf{d}^{(m)})$ не станет меньше чем $w(\mathbf{x}^{(k)})$. Тогда следующее приближение вычисляется, как $\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + \mathbf{d}^{(m)}$. Эта процедура называется демпфированием. В идеальном случае, процедура деления приращения приводит некоторое приближение $\mathbf{x}^{(k)}$ в окрестность точного решения $\mathbf{x}^*$, откуда метод Ньютона сходится уже без дальнейшего демпфирования

Пример 4.6 (метод Ньютона с демпфированием)

Рассмотрим следующую систему уравнений:

$$f_1(x_1, x_2) = \sin\left(\frac{\pi x_1}{20}\right) - x_2 = 0$$

$$f_2(x_1, x_2) = \left(\frac{x_1}{10}\right)^2 - x_2^2 - 1 = 0$$

Результаты вычислений методом Ньютона с начальным приближением $\mathbf{x}^{(0)} = (0.01, 0.01)$ представлены в Таблице 4.6

Таблица 4.6

k	$\mathbf{x}^{(k)}$	$\ \mathbf{x}^{(k)} - \mathbf{x}^{(k-1)}\ _2 / \ \mathbf{x}^{(k)}\ _2$	$\ f(\mathbf{x}^{(k)})\ _2$
1	299.28907 47.01216	9.9996e-01	3.1052e+03
2	149.15142 23.54811	1.0064e+00	7.7636e+02
3	339.03070 -4.95503	5.6628e-01	1.1730e+03
..	...	...	...
8	56.05367 5.45498	8.5945e-01	6.0374e+01
..	...	...	...
11	130.53349 -3.83916	9.1168e-01	1.8419e+02
..	...	...	...
18	7.69136 0.93513	8.9456e-02	4.6604e-01
..	...	...	...
21	5.94612 0.80401	5.4723e-04	2.4848e-07

В данном случае мы наблюдаем немонотонную сходимость и требуется 21 итерация, чтобы получить приближенное решение с заданной точностью. Введение демпфирования в метод Ньютона замечательным образом меняет характер сходимости, как показывает Таблица 4.7.

Таблица 4.7

k	$\mathbf{x}^{(k)}$	$\ \mathbf{x}^{(k)} - \mathbf{x}^{(k-1)}\ _2 / \ \mathbf{x}^{(k)}\ _2$	$\ \mathbf{f}(\mathbf{x}^{(k)})\ _2$
1	4.68624 0.74441	9.9759e-01	2.3773e-01
2	5.94047 0.81741	2.0952e-01	2.5252e-02
3	5.94678 0.80408	2.4591e-03	1.7838e-04
4	5.94612 0.80401	1.1056e-04	9.2859e-09

Более того, шесть операций деления приращения были выполнены только на первой итерации, и это требует меньше вычислений, чем шесть итераций по методу Ньютона. Это говорит о том, что демпфирование не только делает метод Ньютона более надежным, но и может значительно уменьшить количество операций необходимых для получения приближённого решения.

ГЛАВА 5

Численное интегрирование

Список обозначений

$I(f), J(f)$	точное значение интеграла
$e(f)$	ошибка квадратурной формулы
h_k	длина подинтервала при делении отрезка интегрирования
I_k	приближённое значение интеграла, вычисленное с шагом h_k
e_k	оценка ошибки вычисления I_k
N, L, K	число подинтервалов в направлении осей x, y и z соответственно
M	число узлов в квадратурной формуле Гаусса-Кристоффеля
$f(x) \in C^{(n)}[a, b]$	функция $f(x)$ имеет непрерывные производные до порядка n включительно на интервале $[a, b]$
ε_p	заданная точность вычисления интеграла

В этой главе мы рассмотрим методы, которые часто применяются для приближённого вычисления определённых интегралов. Формулы для вычисления одномерных (двумерных) интегралов часто называются квадратурными (кубатурными) формулами. Необходимость вычисления определённых интегралов возникает в различных ситуациях. Самый очевидный случай - когда первообразная подинтегральной функции не может быть выражена через элементарные функции. Если подинтегральная функция задана только в наборе точек, например, она представляет данные эксперимента, то интеграл от этой функции не может быть вычислен аналитическими методами. Формулы приближённого интегрирования также находят широкое применение при создании численных методов для решения дифференциальных и

интегральных уравнений. Всё это говорит о том что численное интегрирование играет важную роль при решении различных практических задач.

В физике и инженерных науках чаще всего встречается интеграл Римана. Он определяется как

$$\int_a^b f(x) dx \equiv \lim_{\max \Delta x_n \rightarrow 0} \sum_{n=1}^N f(x_n^*) \Delta x_n,$$

где x_n^* - произвольная точка из интервала Δx_n и $\bigcup \Delta x_n = [a, b]$,

$$\iint_S f(x, y) dx dy \equiv \lim_{\max \Delta S_n \rightarrow 0} \sum_{n=1}^N f(x_n^*, y_n^*) \Delta S_n,$$

где x_n^*, y_n^* - произвольные точки из подобласти ΔS_n и $\bigcup \Delta S_n = S$,

$$\iiint_V f(x, y, z) dx dy dz \equiv \lim_{\max \Delta V_n \rightarrow 0} \sum_{n=1}^N f(x_n^*, y_n^*, z_n^*) \Delta V_n,$$

где x_n^*, y_n^*, z_n^* - произвольные точки из подобласти ΔV_n и $\bigcup \Delta V_n = V$.

Самый простой способ построения формул приближённого интегрирования непосредственно следует из приведённых выше определений. Если мы будем использовать интервалы конечной длины вместо бесконечно малых интервалов (просто уберём знак $\lim$), то получим простейшие формулы для приближённого вычисления интегралов. Однако такой подход далеко не всегда наиболее эффективный.

5.1 Простейшие квадратурные формулы.

Мы начнем рассмотрение методов для приближенного вычисления определённого интеграла

$$I(f) = \int_a^b f(x) dx \quad (5.1)$$

с метода, основанного на замене подынтегральной функции $f(x)$ некоторым полиномом, интеграл от которого вычисляется легко. Вначале разобьем интервал интегрирования на подинтервалы, вводя множество точек $a = x_0 < x_1 < \dots < x_N = b$. Приближим функцию $f(x)$ на каждом подинтервале $[x_n, x_{n+1}]$ константой $f(x_{n+1/2})$, где $x_{n+1/2} = (x_{n+1} + x_n)/2$ (рисунок 5.1).

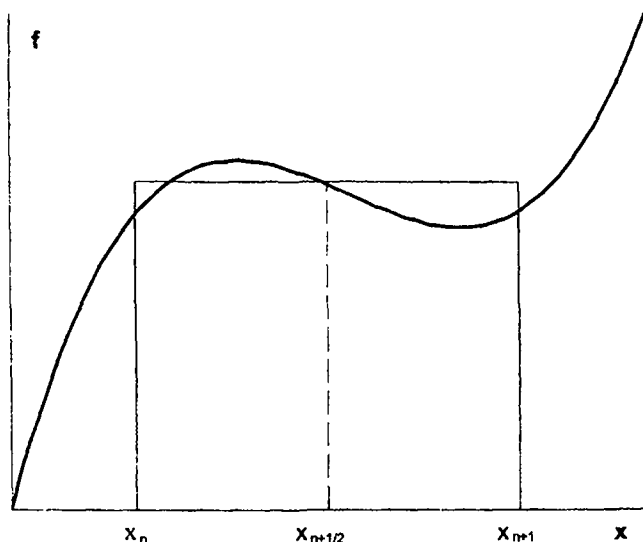


Рисунок 5.1. Приближение функции в методе прямоугольников.

Тогда

$$\int_{x_n}^{x_{n+1}} f(x) dx \approx f(x_{n+1/2})(x_{n+1} - x_n)$$

и сумма

$$I_R(f) = \sum_{n=0}^{N-1} f(x_{n+1/2})(x_{n+1} - x_n) \quad (5.2)$$

даёт приближённое значение определённого интеграла (5.1), то есть

$$I(f) = I_R(f) + e_R(f).$$

Так как сумма (5.2) геометрически представляет собой сумму площадей прямоугольников, то этот метод часто называют методом прямоугольников. Если мы приблизим функцию $f(x)$ на каждом интервале $[x_n$,

$x_{n+1}]$ линейной функцией (рисунок 5.2), то приближённое значение интеграла $I(f)$ есть сумма площадей соответствующих трапеций:

$$I(f) = \sum_{n=0}^{N-1} \frac{1}{2} (f(x_n) + f(x_{n+1}))(x_{n+1} - x_n) + e_T(f) = I_T(f) + e_T(f), \quad (5.3)$$

поэтому этот метод называется методом трапеций.

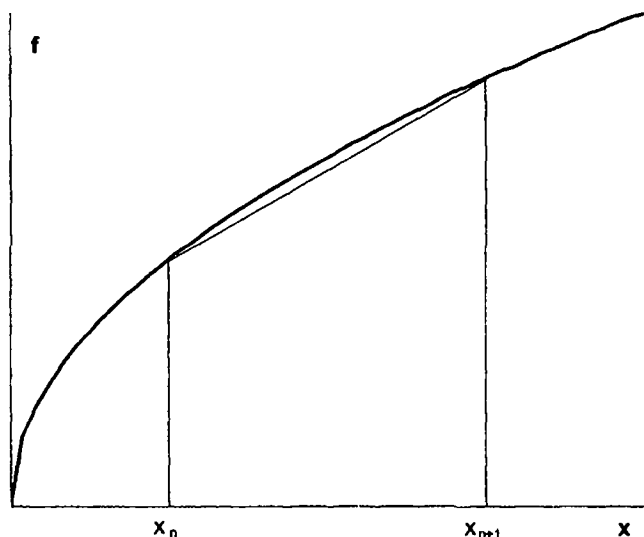


Рисунок 5.2. Приближение функции в методе трапеций.

Пусть $N=2L$, тогда мы можем приблизить подинтегральную функцию $f(x)$ на каждом подинтервале $[x_n, x_{n+1}]$ полиномом второй степени $p(x) = p_2x^2 + p_1x + p_0$ (рисунок 5.3), где коэффициенты p_2 , p_1 , и p_0 определяются из решения системы

$$p(x_{n+1}) = f(x_{n+1})$$

$$p(x_n) = f(x_n)$$

$$p(x_{n-1}) = f(x_{n-1})$$

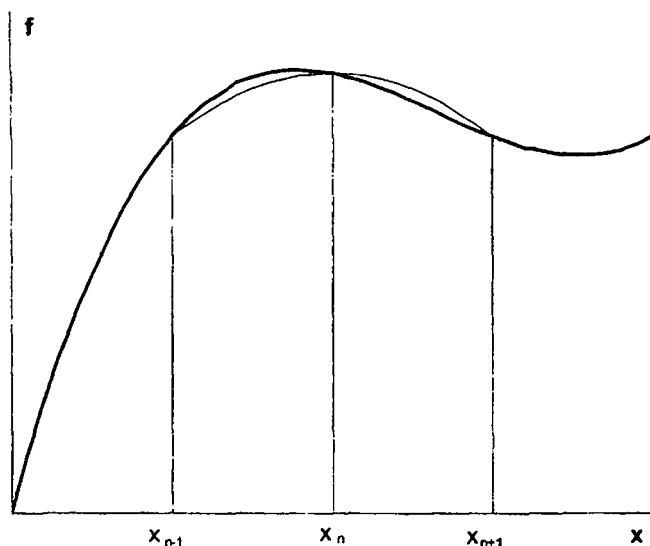


Рисунок 5.3. Приближение функции в методе Симпсона

Тогда следующая сумма даёт приближённое значение интеграла $I(f)$:

$$I(f) = \sum_{m=0}^{L-1} \left[\int_{x_{2m}}^{x_{2m+2}} p(x) dx \right] + e_s(f) =$$

$$\sum_{m=0}^{L-1} \left[\begin{aligned} &a_m f(x_{2m+2}) + \\ &(b_m + c_m - a_m) f(x_{2m+1}) + \\ &c_m f(x_{2m}) \end{aligned} \right] + e_s(f) = I_s(f) + e_s(f) \quad (5.4)$$

где

$$a_m = \frac{2x_{2m+2}^2 - x_{2m}(x_{2m+2} + x_{2m}) - 3x_{2m+1}(x_{2m+2} - x_{2m})}{6(x_{2m+2} - x_{2m+1})}$$

$$b_m = \frac{-2x_{2m}^2 - x_{2m+2}(x_{2m+2} + x_{2m}) - 3x_{2m+1}(x_{2m+2} - x_{2m})}{6(x_{2m+1} - x_{2m})}$$

$$c_m = x_{2m+2} - x_{2m}$$

Этот метод называется методом Симпсона.

Если мы разделим отрезок интегрирования на равные части, то

$$h = x_{n+1} - x_n = \frac{b-a}{N}, \quad x_n = a + nh, \quad n = 0, \dots, N$$

где h называется шагом деления. При этом квадратурные формулы (5.2), (5.3) и (5.4) значительно упрощаются:

$$I_R(f) = h \sum_{n=0}^{N-1} f(x_{n+1/2}) \quad (5.5)$$

$$I_T(f) = \frac{1}{2} h \sum_{n=0}^{N-1} (f(x_n) + f(x_{n+1})) =$$

$$h \left(\frac{1}{2} f(a) + \sum_{n=1}^{N-1} f(x_n) + \frac{1}{2} f(b) \right) \quad (5.6)$$

$$I_S(f) = \frac{1}{3} h \sum_{m=0}^{L-1} (f(x_{2m+2}) + f(x_{2m+1}) + f(x_{2m})) =$$

$$\frac{1}{3} h \left(f(x_0) + 4f(x_1) + 2f(x_2) + \dots + \right.$$

$$\left. 2f(x_{N-2}) + 4f(x_{N-1}) + f(x_N) \right) \quad (5.7)$$

Хотя формулы интегрирования с постоянным шагом имеют простой вид, их использование не всегда приемлемо с точки зрения вычислительной эффективности. Рассмотрим, например, функцию, показанную на рисунке 5.4. Если мы выберем постоянный шаг интегрирования, то он должен быть достаточно малым для того, чтобы более точно интегрировать функцию вблизи пика. Вместо этого можно использовать относительно большой шаг там, где производная функции $f(x)$ мала и меньший шаг в областях резкого изменения функции (рисунок 5.4). Если нам требуется вычислить интеграл с некоторой заданной точностью, то данный приём позволяет это сделать с меньшими затратами, так как значение функции вычисляется в меньшем количестве точек.

Теперь мы обратим наше внимание на оценку ошибок полученных выше квадратурных формул. Для простоты мы рассмотрим случай деления отрезка интегрирования с постоянным шагом. Предположим, что $f(x) \in C^{(4)}[a, b]$, тогда используя формулу Тейлора, представим функцию $f(x)$ на каждом подинтервале $[x_n, x_{n+1}]$ в виде

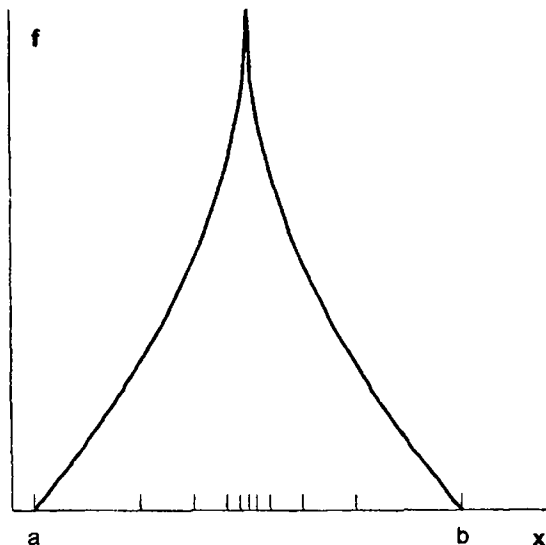


Рисунок 5.4. Ситуация когда следует использовать неравномерное деление отрезка интегрирования.

$$\begin{aligned}
 f(x) &= f(x_{n+1/2}) + f'(x_{n+1/2})(x - x_{n+1/2}) + \\
 &\quad \frac{1}{2}f''(x_{n+1/2})(x - x_{n+1/2})^2 + \frac{1}{6}f'''(x_{n+1/2})(x - x_{n+1/2})^3 + \\
 &\quad \frac{1}{24}f^{(4)}(y)(x - x_{n+1/2})^4
 \end{aligned}$$

$$y \in [x_n, x_{n+1}]$$

Из этого разложения легко получить, что

$$\int_{x_n}^{x_{n+1}} f(x) dx - hf(x_{n+1/2}) = \frac{h^3}{24} f''(x_{n+1/2}) + O(h^5)$$

и, следовательно, выражение для ошибки метода прямоугольников (5.5) принимает вид

$$e_R(f) = \sum_{n=0}^{N-1} \left(\frac{h^3}{24} f''(x_{n+1/2}) + O(h^5) \right).$$

Используя теорему о промежуточном значении, окончательно получим

$$\begin{aligned} e_R(f) &= \frac{(b-a)h^2}{24} \left(\frac{1}{N} \sum_{n=0}^{N-1} f''(x_{n+1/2}) \right) + O(h^4) = \\ &= \frac{(b-a)h^2}{24} f''(y) + O(h^4), \\ &y \in [a, b] \end{aligned}$$

Оценка ошибки для метода трапеций (5.6) и для метода Симпсона (5.7) (в этом случае мы предполагаем, что $f(x) \in C^{(6)}[a, b]$) может быть получена аналогичным образом:

$$\begin{aligned} e_T(f) &= \frac{(b-a)h^2}{12} f''(y) + O(h^4), \quad y \in [a, b], \\ e_S(f) &= \frac{(b-a)h^4}{180} f^{(4)}(y) + O(h^6), \quad y \in [a, b]. \end{aligned}$$

Эти оценки ошибки показывают, что методы прямоугольников и трапеций дают точное значение интеграла от линейных функций. Метод Симпсона даёт точное значение интеграла, если подынтегральная функция является полиномом не выше третьей степени.

Полученные выше выражения для ошибок квадратурных формул можно представить в общем виде как

$$e(f) = Ch^p + O(h^m), \quad (5.8)$$

где C , p и m зависят от метода. Параметр p называется порядком квадратурной формулы, и он показывает как быстро ошибка $e(f)$ стремится к нулю, когда $h \rightarrow 0$. Чем выше порядок квадратурной формулы, тем меньше ошибка приближённого значения интеграла при заданном значении шага h . Однако квадратурные формулы высоких порядков довольно громоздки и не всегда надёжны. Более того, теоретическая оценка ошибки (5.8) справедлива только для функций имеющих непрерывные производные вплоть до порядка p , поэтому формулы высоких порядков имеет смысл использовать для интегрирования очень гладких функций.

5.2 Вычисление интегралов с заданной точностью.

На практике нам нужно вычислить интеграл с некоторой заданной точностью, то есть, должно быть выполнено условие $|e(f)| \leq \epsilon_p$. Как рассматривалось ранее, ошибка зависит от шага, но определить приемлемую величину шага, используя выражение (5.8) и данное выше условие, довольно затруднительно. Оценить ошибку можно непосредственно в процессе вычислений. Обозначим приближённое значение интеграла вычисленного с шагом $h_k = h\alpha^k$ ($0 < \alpha < 1$) как l_k , где h - начальный шаг. Запишем выражение (5.8) для двух последовательных приближений:

$$e_k(f) = l - l_k = Ch_k^p + O(h_k^m)$$

$$e_{k+1}(f) = l - l_{k+1} = C(\alpha h_k)^p + O(h_k^m)$$

Эти два выражения содержат два неизвестных: l и C , и после исключения C мы получим

$$e_{k+1}(f) = l - l_{k+1} = \frac{\alpha^p}{1 - \alpha^p} (l_{k+1} - l_k) + O(h_k^m) \quad (5.9)$$

Так как величина $(h_k)^m$ много меньше, чем $(h_k)^p$, то слагаемым $O((h_k)^m)$ в выражении (5.9) можно пренебречь и, тогда мы получаем возможность оценить ошибку вычисления интеграла через два его приближённых значения. Для того чтобы получить значение l_k , мы должны вычислить подинтегральную функцию в $N_k = (b-a)/h_k$ точках и по мере уменьшения шага h_k это может потребовать значительных вычислительных затрат. Используя два последовательных приближения l_k и l_{k+1} , можно получить более точное приближение к l . Если мы перепишем выражение (5.9) в виде

$$l = l_{k+1} + \frac{\alpha^p}{1 - \alpha^p} (l_{k+1} - l_k) + O(h_k^m) =$$

$$\frac{l_{k+1} - \alpha^p l_k}{1 - \alpha^p} + O(h_k^m) = l_{k+1}^{(e)} + O(h_k^m) \quad (5.10)$$

то $l_{k+1}^{(e)}$ будет ближе к l (так как $m > p$) и это даёт возможность получить приближённое значение интеграла с заданной точностью при относительно большом шаге h_k . Процедура (5.9) и (5.10) называется экстраполяцией по Ричардсону.

Суммируя всё изложенное выше, можно предложить следующую схему для вычисления интегралов:

- 1) зададим начальное значение шага h и параметр α , затем вычислим I_0 и положим

$$I_0^{(e)} = I_0.$$

- 2) для каждого значения $k=1, 2, \dots$ вычислим I_k с шагом $h_k = h\alpha^k$ и улучшенное приближение

$$I_k^{(e)} = \frac{I_k - \alpha^p I_{k-1}}{1 - \alpha^p}$$

- 3) если

$$\frac{\alpha^p}{1 - \alpha^p} \left| \frac{I_k - I_{k-1}}{I_k} \right| \leq \varepsilon_p$$

или

$$\left| \frac{I_k^{(e)} - I_{k-1}^{(e)}}{I_k^{(e)}} \right| \leq \varepsilon_p,$$

тогда процесс вычисления завершается и

$$\left| \frac{I - I_k}{I} \right| \leq \varepsilon_p$$

или

$$\left| \frac{I - I_k^{(e)}}{I} \right| \leq \varepsilon_p$$

Пример 5.1 (применение экстраполяции по Ричардсону)

Рассмотрим следующий интеграл

$$I = \int_0^1 x \ln(1+x) dx = 0.25$$

Для вычисления этого интеграла мы воспользуемся методом трапеций (5.6) ($p=2$, $m=4$). Выберем начальный шаг $h=1$ и будем последовательно уменьшать его в два раза ($\alpha=0.5$). Результаты вычислений представлены в Таблице 5.1.

Таблица 5.1

k	N _k	I _k	e _k из (5.9)	I _k ^(e)	$\left \frac{I_k^{(e)} - I_{k-1}^{(e)}}{I_k^{(e)}} \right $
0	1	0.346574		0.346574	
1	2	0.274653	8.7287e-02	0.250700	3.8254e-01
2	4	0.256201	2.4007e-02	0.250050	2.5167e-03
3	8	0.251553	6.1594e-03	0.250003	1.8778e-04
4	16	0.250388	1.5501e-03	0.250000	1.2441e-05
5	32	0.250097	3.8818e-04		
6	64	0.250024	9.7085e-05		
7	128	0.250006	2.4274e-05		
8	256	0.250002	6.0686e-06		
9	512	0.250000	1.5172e-06		

Этот пример демонстрирует эффективность применения экстраполяции по Ричардсону. Улучшенное значение интеграла $I^{(e)}$ приближается к I быстрее чем I_k , что значительно уменьшает количество вычислений.

Существуют ситуации, когда подынтегральная функция непрерывна и ограничена на интервале интегрирования, а её производные имеют особые точки на этом интервале. В этом случае константа C в (5.8) стремиться к бесконечности, когда $h \rightarrow 0$. Интеграл

$$I = \int_0^1 \sqrt{x} \, dx$$

является примером такой ситуации. С другой стороны понятно, что интеграл от непрерывной ограниченной функции может быть вычислен с любой точностью в рамках машинной арифметики. Это означает, что поведение ошибки описывается выражением (5.8), но параметры C , p и m неизвестны и, более того, они зависят от конкретной подынтегральной функции. В этом случае экстраполяция по Ричардсону строится на основе трёх последовательных приближений:

$$I = I_k + Ch_k^p + O(h_k^m)$$

$$I = I_{k+1} + C(\alpha h_k)^p + O(h_k^m)$$

$$I = I_{k+2} + C(\alpha^2 h_k)^p + O(h_k^m)$$

После исключения из этих выражений неизвестных C и p , мы получим

$$e_{k+2}(f) = I - I_{k+2} = I_k - I_{k+2} + \frac{(I_k - I_{k+1})^2}{2I_{k+1} - I_k - I_{k+2}} + O(h_k^m) \quad (5.11)$$

и улучшенное приближение вычисляется по формуле

$$I = \frac{I_{k+1}^2 - I_k I_{k+2}}{2I_{k+1} - I_k - I_{k+2}} + O(h_k^m) = I_{k+2}^{(e)} + O(h_k^m) \quad (5.12)$$

Что интересно, это выражение совпадает с выражением (1.14), поэтому формулу (5.12) часто называют процессом Эйткена для интегралов. Порядок квадратурной формулы, используемой для вычисления I_k , определяется как

$$p = \lim_{k \rightarrow \infty} \frac{\ln \frac{I_{k+2} - I_{k+1}}{I_{k+1} - I_k}}{\ln \alpha}.$$

Теперь вычислительная схема конструируется следующим образом:

- 1) зададим начальное значение шага h и параметр α , затем вычислим I_0 и I_1 , и положим $I_1^{(e)} = I_1$.
- 2) для каждого значения $k = 2, 3, \dots$ вычислим I_k с шагом $h_k = h\alpha^k$ и улучшенное приближение и улучшенное приближение по формуле (5.12)
- 3) если $|e_k(f)| \leq \varepsilon_p$ (смотри (5.11)) или

$$\left| \frac{I_k^{(e)} - I_{k-1}^{(e)}}{I_k^{(e)}} \right| \leq \varepsilon_p,$$

тогда процесс вычисления завершается и

$$\left| \frac{I - I_k}{I} \right| \leq \varepsilon_p$$

или

$$\left| \frac{I - I_k^{(e)}}{I} \right| \leq \varepsilon_p$$

Пример 5.2 (применение экстраполяции по Ричардсону, когда производные подинтегральной функции имеют особенность на интервале интегрирования)

Рассмотрим интеграл

$$I = \int_0^1 \sqrt{1-x^2} \, dx = \frac{\pi}{4} \approx 0.785398.$$

Производные подинтегральной функции:

$$f'(x) = -x(1-x^2)^{-\frac{1}{2}}, \quad f''(x) = -(1-x^2)^{-\frac{3}{2}},$$

и они имеют особенность в точке $x=1$. Для вычисления этого интеграла мы воспользуемся методом трапеций (5.6). Выберем начальный шаг $h=1$ и будем последовательно уменьшать его в два раза ($\alpha=0.5$). Результаты вычислений представлены в Таблице 5.2.

Таблица 5.2

k	N_k	I_k	e_k из (5.11)
0	1	0.500000	
1	2	0.683013	
2	4	0.748927	1.32e-01
3	8	0.772455	4.32e-02
4	16	0.780813	1.48e-02
5	32	0.783776	5.19e-03
6	64	0.784824	1.83e-03
..	...		
10	1024	0.785390	2.85e-05

Таблица 5.2 (продолжение)

k	N _k	I _k ^(e) из (5.12)	$\frac{ I_k^{(e)} - I_{k-1}^{(e)} }{I_k^{(e)}}$	I _k ^(e) из (5.10)
0	1			0.500000
1	2	0.683013		0.744017
2	4	0.786031	1.31e-01	0.770899
3	8	0.785514	6.57e-04	0.780297
4	16	0.785413	1.21e-04	0.783599
5	32	0.785402	2.17e-05	0.784763
6	64	0.785399	3.87e-06	0.785174
..	...			
10	1024			0.785395

И опять значения $I_k^{(e)}$, основанное на экстраполяции (5.12), дают лучшее приближение к I по сравнению с I_k . Порядок метода трапеций для данного интеграла можно оценить следующим образом:

$$p \approx -\frac{\ln \frac{I_6 - I_5}{I_5 - I_4}}{\ln 2} \approx 1.5$$

Результаты последнего столбца Таблицы 5.2 показывают, что экстраполяция (5.10), основанная на теоретическом значении порядка метода трапеций $p=2$, даёт лишь незначительное улучшение сходимости.

5.3. Формулы Гаусса-Кристоффеля.

В этом параграфе мы рассмотрим вычисление интегралов специального вида

$$K(f) = \int_c^d \rho(x)f(x)dx, \quad (5.13)$$

где $\rho(x)$ называется весовой функцией. До сих пор в этой главе мы имели дело с квадратурными формулами, в которых подинтегральная функция вычислялась в заданных точках (узлах). Теперь мы обсудим

некоторые специальные формулы для интеграла (5.13) которые имеют следующий вид (формулы Гаусса-Кристоффеля):

$$I(f) = \sum_{m=1}^M w_m f(z_m) + e_M(f), \quad (5.14)$$

где веса $w_1, \dots, w_M$ и узлы $z_1, \dots, z_M$ свободные параметры. Следует заметить, что весовая функция $\rho(x)$ не участвует в вычислениях явным образом. Как мы увидим позднее, она влияет на выбор весов и узлов в формуле (5.14). Наша цель состоит в определении этих параметров таким образом, чтобы квадратурная формула (5.14) имела наименьшую возможную погрешность. Так как теперь мы имеем $2M$ параметров w_m и z_m , то можно построить формулу, которая даёт точное значение интеграла ($e_M(f)=0$) от всех полиномов степени не выше чем $2M-1$. Поясним это утверждение на примере. Пусть функция $f(x)$ есть полином третьей степени: $f(x) = a_3x^3 + a_2x^2 + a_1x + a_0$, и $[c,d]=[0,1]$. Пусть также $\rho(x)=1$ и $M=2$. Предположим, что квадратурная формула (5.14) даёт точное значение интеграла (5.13). Тогда мы имеем

$$I = a_0 + \frac{1}{2}a_1 + \frac{1}{3}a_2 + \frac{1}{4}a_3 = w_1(a_0 + a_1z_1 + a_2z_1^2 + a_3z_1^3) + w_2(a_0 + a_1z_2 + a_2z_2^2 + a_3z_2^3)$$

Перепишем это выражение в виде

$$a_0(1 - w_1 - w_2) + a_1\left(\frac{1}{2} - w_1z_1 - w_2z_2\right) + a_2\left(\frac{1}{3} - w_1z_1^2 - w_2z_2^2\right) + a_3\left(\frac{1}{4} - w_1z_1^3 - w_2z_2^3\right) = 0$$

Решение системы уравнений

$$\begin{aligned} 1 - w_1 - w_2 &= 0 \\ \frac{1}{2} - w_1z_1 - w_2z_2 &= 0 \\ \frac{1}{3} - w_1z_1^2 - w_2z_2^2 &= 0 \\ \frac{1}{4} - w_1z_1^3 - w_2z_2^3 &= 0 \end{aligned}$$

даёт веса w_1, w_2 и узлы z_1, z_2 квадратурной формулы, которая точна для полиномов третьей степени.

Возвратимся теперь к интегралу (5.13). Если

- 1) $\rho(x) \geq 0$ на интервале $[c, d]$ и $\rho(x) \in C^{(0)}[c, d]$,
- 2) $\int_c^d \rho(x) dx$ существует,

тогда существует набор полиномов $P_n(x)$ ортогональных на отрезке $[c, d]$ с весовой функцией $\rho(x)$, то есть

$$\int_c^d \rho(x) P_n(x) P_m(x) dx = \begin{cases} 0, & n \neq m \\ \text{const}, & n = m \end{cases} \quad (5.15)$$

Все нули этих полиномов вещественны и расположены внутри отрезка $[c, d]$. Составим полином степени M по узлам квадратурной формулы:

$$Q_M(x) = \prod_{m=1}^M (x - z_m)$$

Пусть $f(x) = Q_M(x)P_m(x)$ и если $m \leq M-1$, то функция $f(x)$ есть полином степени не выше $2M-1$. Следовательно, формула (5.14) точна для этой функции. Это даёт

$$\int_c^d \rho(x) Q_M(x) P_m(x) dx = \sum_{m=1}^M w_m Q_M(z_m) P_m(z_m) = 0, \quad (5.16)$$

$m = 1, \dots, M-1$

так как $Q_M(z_m) = 0$. Значит, $Q_M(x)$ ортогонален ко всем полиномам $P_m(x)$ степени $m \leq M-1$. Разложим теперь $Q_M(x)$ по полиномам $P_n(x)$:

$$Q_M(x) = \sum_{n=1}^M b_n P_n(x).$$

Подставим это выражение в (5.16) и, учитывая (5.17) получим

$$\int_c^d \rho(x) Q_M(x) P_m(x) dx =$$

$$\sum_{n=1}^M b_n \int_c^d \rho(x) P_n(x) P_m(x) dx = b_m \cdot \text{const} = 0,$$

$$m = 1, \dots, M-1$$

то есть $b_m=0$ когда $m \leq M-1$. Тогда согласно нашему разложению, $Q_M(x)$ совпадает с $P_M(x)$:

$$Q_M(x) = \prod_{m=1}^M (x - z_m) = b_M P_M(x)$$

Следовательно, узлы квадратурной формулы (нули полинома $Q_M(x)$) являются нулями $P_M(x)$. Получим выражение для весов квадратурной формулы. Составим полином

$$S_n(x) = \prod_{m=1, m \neq n}^M \frac{(x - z_m)}{(z_n - z_m)}.$$

Это полином степени $M-1$ и формула (5.14) для него точна. Учитывая, что $S_n(z_m) \neq 0$ когда $m \neq n$, получим

$$\int_c^d \rho(x) S_n(x) dx = \sum_{m=1}^M w_m S_n(z_m) = w_n,$$

$$n = 1, \dots, M$$

Для некоторых интегралов вида (5.13), узлы и веса квадратурной формулы (5.14) известны, например, для следующих случаев, часто встречающихся на практике:

1) $\rho(x) = 1$, $c = -1$, $d = 1$ (формула Гаусса)

2) $\rho(x) = x^k$, $c = 0$, $d = 1$

3) $\rho(x) = \sqrt{1-x}$, $c = 0$, $d = 1$

$$4) \rho(x) = \frac{1}{\sqrt{1-x}}, c=0, d=1$$

$$5) \rho(x) = \frac{1}{\sqrt{1-x^2}}, c=-1, d=1$$

$$6) \rho(x) = \sqrt{1-x^2}, c=-1, d=1 \quad (5.17)$$

$$7) \rho(x) = \sqrt{\frac{x}{1-x}}, c=0, d=1$$

$$8) \rho(x) = -\ln(x), c=0, d=1$$

$$9) \rho(x) = \exp(-x), c=0, d=\infty$$

$$10) \rho(x) = \exp(-x^2), c=-\infty, d=\infty$$

Выражение для ошибки квадратурных формул (5.14) можно представить в виде

$$e_M(f) = C(M)f^{(2M)}(y), y \in [c, d],$$

и они являются наиболее точными формулами для вычисления интегралов от гладких функций. Более подробную информацию о вычислении интеграла (5.13) с весовыми функциями (5.17) можно найти в Приложении А. Это приложение включает в себя информацию о соответствующих ортогональных полиномах, табличные данные о весах и узлах квадратурных формул, а также значения констант ошибки $C(M)$.

Контроль ошибки формул Гаусса-Кристоффеля осуществляется путём выбора подходящего количества узлов интегрирования (чем больше узлов, тем меньше константа $C(M)$). Когда $\rho(x)=1$ квадратура Гаусса может быть применена к интегралу по конечному отрезку $[a, b]$ (с помощью преобразования $x=(b+a)/2+z(b-a)/2$):

$$\int_a^b f(x) dx = \frac{b-a}{2} \int_{-1}^1 f\left(\frac{b+a}{2} + \frac{b-a}{2} z\right) dz =$$

$$\frac{b-a}{2} \sum_{m=1}^M w_m f\left(\frac{b+a}{2} + \frac{b-a}{2} z_m\right) + e_M(f)$$

В этом случае используется смешанный подход к контролю ошибки. Если мы разделим интервал интегрирования $[a, b]$ на отрезки длиной h , то можно построить следующую формулу интегрирования:

$$\int_a^b f(x) dx = \frac{1}{2} h \sum_{n=0}^{N-1} \left[\sum_{m=1}^M w_m f\left(x_{n+1/2} + \frac{1}{2} h z_m\right) \right] + e_N(f),$$

$$x_{n+1/2} = a + \left(n + \frac{1}{2}\right) h, \quad h = \frac{b-a}{N}, \quad (5.18)$$

$$n = 0, \dots, N-1$$

Выражение для ошибки этой формулы имеет вид

$$e_N(f) = \frac{(b-a)h^{2M}}{2^{2M+1}} C(M) f^{(2M)}(y), \quad y \in [a, b],$$

и теперь ошибка зависит не только от числа узлов в каждом под-интервале $[x_n, x_{n+1}]$, но и от шага h .

Пример 5.3 (формула Гаусса)

Рассмотрим интеграл из примера 5.1:

$$I = \int_0^1 x \ln(1+x) dx = 0.25,$$

где подинтегральная функция $f(x) \in C^{(\infty)}[0, 1]$. Используем квадратурную формулу (5.18) ($N=1$, $h=1$) с тремя узлами и весами ($M=3$) и в данном случае она примет вид

$$I \approx I_1^{(3)} = \frac{1}{2} \sum_{m=1}^3 w_m \left(\frac{1}{2} + \frac{1}{2} z_m \right) \ln \left(\frac{3}{2} + \frac{1}{2} z_m \right),$$

где w_m и z_m равны (смотри Таблицу A1 в Приложении A):

$$z_1 = -0.774797, \quad w_1 = 0.555556$$

$$z_2 = 0, \quad w_2 = 0.888889$$

$$z_3 = 0.774797, \quad w_3 = 0.555556$$

Вычисление даёт следующий результат:

$$I_1^{(3)} \approx 0.249992$$

с абсолютной ошибкой

$$e = 1 - I_1^{(3)} \approx 7.62 \cdot 10^{-6}.$$

Этот пример демонстрирует, что формула Гаусса позволяет вычислять интегралы с высокой точностью, даже если мы используем всего несколько узлов.

5.4 Несобственные интегралы.

В этом параграфе мы обсудим применение рассмотренных выше квадратурных формул к вычислению несобственных интегралов. Начнём с темы

5.4.1 Бесконечные пределы интегрирования

Как было рассмотрено в параграфе 5.3, формулы Гаусса-Кристоффеля могут быть использованы для вычисления интегралов с весовыми функциями $\exp(-x)$ и $\exp(-x^2)$ на интервалах $[0, \infty)$ и $(-\infty, \infty)$, соответственно. Мы всегда можем ввести эти весовые функции искусственным образом, сделав простое преобразование:

$$\int_0^{\infty} f(x) dx = \int_0^{\infty} \exp(-x) (\exp(x)f(x)) dx$$

или

$$\int_{-\infty}^{\infty} f(x) dx = \int_{-\infty}^{\infty} \exp(-x^2) (\exp(x^2)f(x)) dx.$$

Теперь применим формулу (5.14) и получим следующие выражения:

$$\int_0^{\infty} \exp(-x) (\exp(x)f(x)) dx \approx \sum_{m=1}^M w_m^{(L)} \exp(z_m^{(L)}) f(z_m^{(L)})$$

или

$$\int_{-\infty}^{\infty} \exp(-x^2) (\exp(x^2) f(x)) dx \approx \sum_{m=1}^M w_m^{(H)} \exp((z_m^{(H)})^2) f(z_m^{(H)})$$

Значения модифицированных весов

$$w_m^{(L)} \exp(z_m^{(L)}) \text{ и } w_m^{(H)} \exp((z_m^{(H)})^2)$$

представлены в Таблицах A10 и A11 Приложения А. Другой способ обращения с бесконечным интервалом интегрирования - преобразовать его в конечный интервал. Замены переменных $y=x/(1+x)$ или $y=\exp(-x)$ преобразуют интервал $[0, \infty)$ в интервал $[0, 1]$. Аналогично, замена переменной $y=(\exp(x)-1)/(\exp(x)+1)$ преобразует интервал $(-\infty, \infty)$ в интервал $[-1, 1]$. Если преобразованная подынтегральная функция непрерывна и ограничена, то мы получаем стандартную проблему интегрирования. В противном случае мы сталкиваемся с проблемой интегрирования функции имеющей особенность.

Пример 5.4 (преобразование полубесконечного интервала интегрирования)

Рассмотрим следующий интеграл:

$$I = \int_0^{\infty} \frac{dx}{1+x^2}.$$

Используя преобразование $y=x/(1+x)$ получим

$$I = \int_0^1 \frac{dy}{y^2 + (1-y)^2}.$$

Вычисление этого интеграла уже не составляет особого труда.

5.4.2 Подынтегральная функция имеет особенность

В параграфе 5.3 мы рассматривали интегралы вида (5.13). Весовые функции и интервалы (4), (5), (7) и (8) представленные в (5.17), являются часто встречающимися на практике особенностями. Поэтому квадратурные формулы (5.14) с соответствующими узлами и весами могут быть использованы для численного интегрирования функций с этими особенностями.

Пример 5.5 (применение формулы Гаусса-Кристоффеля к вычислению интеграла от функции с особенностью)

Рассмотрим следующий интеграл:

$$I = \int_0^1 \frac{\ln(x)}{1+x} dx = -\frac{\pi^2}{12} \approx -0.822467,$$

с особенностью $\ln(x)$ в точке $x=0$. Будем использовать формулу (5.14) с двумя узлами и весами ($M=2$) и для данного интеграла она имеет вид

$$I \approx I_2 = -\sum_{m=1}^2 \frac{w_m}{1+z_m},$$

где w_m и z_m равны (смотри Таблицу A9 в Приложении A):

$$z_1 = 0.112009, w_1 = 0.718539$$

$$z_2 = 0.602277, w_2 = 0.281461$$

Вычисление даёт следующий результат: $I_2 \approx -0.821826$, с абсолютной ошибкой $e_2 = |I - I_2| \approx 6.4 \cdot 10^{-4}$. Этот пример демонстрирует, что формулы Гаусса-Кристоффеля позволяют довольно точно вычислять интегралы от функций с особенностями, даже если используется всего несколько узлов.

Другой способ интегрирования функций с особенностями требует некоторых предварительных преобразований исходного интеграла. Если мы можем выделить особенность и затем вычислить интеграл от неё точно или приближённо, тогда проблема сводится к вычислению обыкновенного интеграла. В качестве примера рассмотрим

$$\begin{aligned} \int_0^1 \frac{\exp(-x)}{\sqrt{x}} dx &= \int_0^1 \frac{dx}{\sqrt{x}} + \int_0^1 \frac{\exp(-x)-1}{\sqrt{x}} dx = \\ &= 2 + \int_0^1 \frac{\exp(-x)-1}{\sqrt{x}} dx \end{aligned}$$

Так как теперь функция $f(x) = (\exp(-x)-1)x^{1/2}$ в точке $x=0$ равна нулю и непрерывна на интервале $[0, 1]$, то для вычисления последнего интеграла можно использовать стандартные формулы интегрирования.

5.5 Многомерное интегрирование.

При переходе от задачи вычисления одномерных интегралов к многомерному интегрированию возникает целый ряд новых проблем. В одномерном случае мы имеем три возможных типа области интегрирования: конечный, полубесконечный и бесконечный интервалы. В многомерном случае мы сталкиваемся с большим разнообразием областей интегрирования. В дополнение, подинтегральная функция может иметь особенность не только в точке, а также во множествах точек таких как, например, интервал, плоскость т.д. Поэтому вычисление многомерных интегралов представляет собой значительно более сложную задачу по сравнению с одномерным случаем.

Для таких областей как квадрат, куб, цилиндр и т.д., которые представляют собой декартовы произведения областей более низкой размерности, можно построить формулы интегрирования «перемножая» формулы для вычисления интегралов более низкой размерности (правило перемножения). Так если

$$\int_a^b f(x) dx \approx \sum_{n=1}^N w_n f(x_n)$$

есть одномерная формула интегрирования, то

$$\int_a^b \int_a^b f(x, y) dx dy \approx \sum_{n=1}^N \sum_{m=1}^L w_n w_m f(x_n, x_m),$$

даёт формулу для вычисления двумерного интеграла. Однако такие методы не всегда самые эффективные. В этом параграфе мы рассмотрим более универсальный подход к вычислению многомерных интегралов, который, в частности, включает в себя и правило перемножения.

Вначале рассмотрим интегрирование по квадратной области $S = \{-1 \leq r \leq 1, -1 \leq s \leq 1\}$ на плоскости, и наша задача - построить кубатурную формулу для вычисления интеграла

$$J = \int_{-1}^1 \int_{-1}^1 f(r, s) dr ds \quad (5.19)$$

Введём некоторые узлы (r_1, s_1) , (r_2, s_2) , ..., (r_L, s_L) , и соответствующие веса $w_1, \dots, w_L$ таким образом, чтобы формула

$$J \approx \sum_{k=1}^L w_k f(r_k, s_k) \quad (5.20)$$

была точной для одночленов вида $r^m s^m$, $n \geq 0$, $m \geq 0$, как можно более высокой степени $n+m \leq p$. Для простоты мы рассмотрим случай $L=4$ (рисунок 5.5).

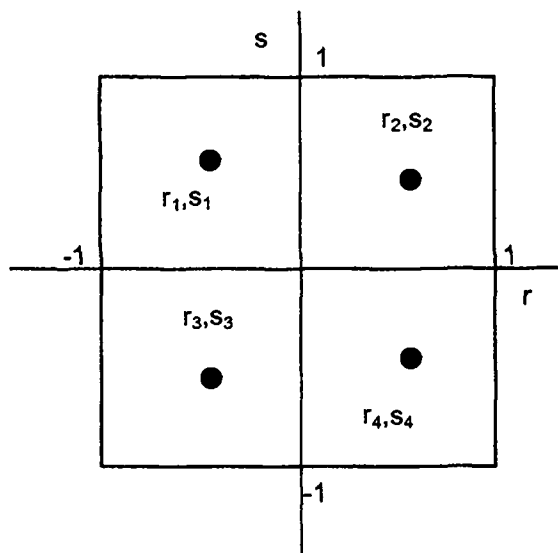


Рисунок 5.5. Расположение узлов кубатурной формулы (5.20) в случае $L=4$.

Подставляя функцию $f(r,s)=r^m s^m$ в формулу (5.20), получим

$$w_1 + w_2 + w_3 + w_4 = \int_{-1}^1 \int_{-1}^1 dr ds = 4,$$

$$w_1 r_1 + w_2 r_2 + w_3 r_3 + w_4 r_4 = \int_{-1}^1 \int_{-1}^1 r dr ds = 0,$$

$$\begin{aligned}
w_1 s_1 + w_2 s_2 + w_3 s_3 + w_4 s_4 &= \int_{-1}^1 \int_{-1}^1 s \, dr \, ds = 0, \\
w_1 r_1 s_1 + w_2 r_2 s_2 + w_3 r_3 s_3 + w_4 r_4 s_4 &= \int_{-1}^1 \int_{-1}^1 rs \, dr \, ds = 0, \quad (5.21) \\
w_1 r_1^2 + w_2 r_2^2 + w_3 r_3^2 + w_4 r_4^2 &= \int_{-1}^1 \int_{-1}^1 r^2 \, dr \, ds = \frac{4}{3}, \\
w_1 s_1^2 + w_2 s_2^2 + w_3 s_3^2 + w_4 s_4^2 &= \int_{-1}^1 \int_{-1}^1 s^2 \, dr \, ds = \frac{4}{3},
\end{aligned}$$

и так далее.

Если мы наложим некоторые условия на расположение узлов, то эти уравнения можно упростить. Предположим, что $(r_1, s_1) = (-\beta, \gamma)$, $(r_2, s_2) = (\beta, \gamma)$, $(r_3, s_3) = (\beta, -\gamma)$ и $(r_4, s_4) = (-\beta, -\gamma)$, тогда система (5.21) принимает вид

$$\begin{aligned}
w_1 + w_2 + w_3 + w_4 &= 4, \\
\beta(-w_1 + w_2 + w_3 - w_4) &= 0, \\
\gamma(w_1 + w_2 - w_3 - w_4) &= 0, \\
\beta\gamma(-w_1 + w_2 - w_3 + w_4) &= 0, \\
\beta^2(w_1 + w_2 + w_3 + w_4) &= 4/3, \\
\gamma^2(w_1 + w_2 + w_3 + w_4) &= 4/3.
\end{aligned}$$

Теперь мы имеем шесть уравнений для шести неизвестных и решение этой системы: $\beta = \gamma = (1/3)^{1/2}$ и $w_k = 1$, $k=1, \dots, 4$. Тогда кубатурная формула (5.20) приобретает вид

$$\begin{aligned}
J &\approx f(-\beta, \beta) + f(\beta, \beta) + f(\beta, -\beta) + f(-\beta, -\beta), \\
\beta &= \sqrt{1/3}
\end{aligned} \quad (5.22)$$

Это в точности даёт двумерную формулу Гаусса, которая может быть также получена по правилу перемножения для $M=2$. Мы можем расположить узлы и некоторым другим способом, например, $(r_1, s_1) = (-\beta, 0)$, $(r_2, s_2) = (0, \beta)$, $(r_3, s_3) = (\beta, 0)$ и $(r_4, s_4) = (0, -\beta)$. Решая уравнения (5.21), мы получим $\beta = (2/3)^{1/2}$ и $w_k = 1$, $k=1, \dots, 4$, и теперь кубатурная формула (5.20) записывается в виде

$$J \approx f(-\beta, 0) + f(0, \beta) + f(\beta, 0) + f(0, -\beta), \quad (5.23)$$

$$\beta = \sqrt{2/3}$$

На основе полученных выше формул, построим формулы интегрирования по произвольной прямоугольной области $S = \{a \leq x \leq b, c \leq y \leq d\}$. Вначале разобьём область S на ячейки, вводя сетку с узлами (x_n, y_m) , которая задаётся следующим образом:

$$x_n = a + nh_x, h_x = \frac{b-a}{N}, n = 0, \dots, N$$

$$y_m = c + mh_y, h_y = \frac{d-c}{L}, m = 0, \dots, L$$

где h_x и h_y определяют размер каждой ячейки $S = \{x_n \leq x \leq x_{n+1}, y_m \leq y \leq y_{m+1}\}$. Используя формулы, полученные для интеграла (5.19) и сделав замену переменных $x = 0.5(x_n + x_{n+1}) + 0.5h_x r$ и $y = 0.5(y_m + y_{m+1}) + 0.5h_y s$, мы сначала вычислим интегралы по каждой ячейке S_{nm} , а затем просуммируем все полученные значения. В результате, мы придём к следующим формулам:

$$I = \int_c^d \int_a^b f(x, y) dx dy =$$

$$\frac{1}{4} h_x h_y \sum_{n=0}^{N-1} \sum_{m=0}^{L-1} \begin{bmatrix} f(x_{n+1/2} - 0.5\beta h_x, y_{m+1/2} + 0.5\beta h_y) + \\ f(x_{n+1/2} + 0.5\beta h_x, y_{m+1/2} + 0.5\beta h_y) + \\ f(x_{n+1/2} + 0.5\beta h_x, y_{m+1/2} - 0.5\beta h_y) + \\ f(x_{n+1/2} - 0.5\beta h_x, y_{m+1/2} - 0.5\beta h_y) \end{bmatrix} + e(f) \quad (5.24)$$

на основе кубатуры (5.22) и

$$I = \int_c^d \int_a^b f(x, y) dx dy =$$

$$\frac{1}{4} h_x h_y \sum_{n=0}^{N-1} \sum_{m=0}^{L-1} \begin{bmatrix} f(x_{n+1/2} - 0.5\beta h_x, y_{m+1/2}) + \\ f(x_{n+1/2}, y_{m+1/2} + 0.5\beta h_y) + \\ f(x_{n+1/2} + 0.5\beta h_x, y_{m+1/2}) + \\ f(x_{n+1/2}, y_{m+1/2} - 0.5\beta h_y) \end{bmatrix} + e(f) \quad (5.25)$$

на основе кубатуры (5.23). Точка $(x_{n+1/2}, y_{m+1/2})$ обозначает центр ячейки и имеет координаты $x_{n+1/2} = x_n + 0.5h_x$ и $y_{m+1/2} = y_m + 0.5h_y$. Формулы (5.22) и (5.23) точны для всех одночленов вида $x^m y^n$, $n \geq 0$, $m \geq 0$, $n+m \leq 3$. Тогда выражение для ошибки формул (5.24) и (5.25) можно записать в виде

$$e(f) \sim C(\max(h_x, h_y))^4,$$

где константа C зависит от частных производных функции $f(x, y)$ и мы предполагаем, что все эти производные непрерывны. Для того чтобы вычислить интеграл с заданной точностью мы можем использовать метод представленный в параграфе 5.2, только теперь l_k будет обозначать приближённое значение интеграла, вычисленное при делении области интегрирования на $N_k \times L_k$ ячеек, где

$$N_k = \frac{b-a}{h_x^{(k)}}, h_x^{(k)} = h_x \alpha^k, 0 < \alpha < 1$$

$$L_k = \frac{d-c}{h_y^{(k)}}, h_y^{(k)} = h_y \alpha^k$$

представляют собой число подинтервалов в направлениях x и y , соответственно.

Пример 5.6 (вычисление двумерного интеграла)

Рассмотрим следующий интеграл

$$I = \int_0^1 \int_0^1 \frac{dx dy}{1+xy} = \frac{\pi^2}{12} \approx 0.822467.$$

Для вычисления этого интеграла мы воспользуемся кубатурной формулой (5.24). Выберем начальный шаг $h_x = h_y = h = 1$ и будем последовательно уменьшать его в два раза ($\alpha = 0.5$). Результаты вычислений представлены в Таблице 5.3.

Таблица 5.3

k	Число ячеек	l_k	e_k из (5.9)	$\frac{ I - l_k }{I}$
0	1	0.822014		5.5072e-04
1	4	0.822432	3.3877e-05	4.2592e-05
2	16	0.822465	2.6510e-06	2.8275e-06

Этот пример показывает, что приближённые значения интеграла, вычисленные по формуле (5.24) сходятся очень быстро к точному значению интеграла.

Используя рассмотренный выше подход, можно построить формулы интегрирования и для интегралов более высокой размерности. Например, рассмотрим трёхмерный интеграл

$$J = \int_{-1}^1 \int_{-1}^1 \int_{-1}^1 f(r, s, t) dr ds dt.$$

В этом случае, трёхмерный аналог кубатурной формулы (5.25) имеет вид

$$J \approx \frac{8}{6} \begin{pmatrix} f(-1, 0, 0) + f(1, 0, 0) + f(0, -1, 0) + \\ f(0, 1, 0) + f(0, 0, -1) + f(0, 0, 1) \end{pmatrix}.$$

Тогда формула интегрирования по прямоугольному параллелепипеду записывается в следующем виде:

$$I = \int_a^b \int_c^d \int_o^p f(x, y, z) dx dy dz = \frac{1}{6} h_x h_y h_z \sum_{k=0}^{K-1} \sum_{n=0}^{N-1} \sum_{m=0}^{L-1} \begin{bmatrix} f(x_n, y_{m+1/2}, z_{k+1/2}) + \\ f(x_{n+1}, y_{m+1/2}, z_{k+1/2}) + \\ f(x_{n+1/2}, y_m, z_{k+1/2}) + \\ f(x_{n+1/2}, y_{m+1}, z_{k+1/2}) + \\ f(x_{n+1/2}, y_{m+1/2}, z_k) + \\ f(x_{n+1/2}, y_{m+1/2}, z_{k+1}) \end{bmatrix} + e(f) \quad (5.26)$$

где

$$x_n = a + nh_x, h_x = \frac{b-a}{N}, n = 0, \dots, N$$

$$y_m = c + mh_y, h_y = \frac{d-c}{L}, m = 0, \dots, L,$$

$$z_k = o + kh_z, h_z = \frac{p-o}{K}, k = 0, \dots, K$$

и

$$x_{n+1/2} = x_n + \frac{1}{2} h_x$$

$$y_{m+1/2} = y_m + \frac{1}{2} h_y ,$$

$$z_{k+1/2} = z_k + \frac{1}{2} h_z$$

До сих пор мы обсуждали интегрирование по прямоугольным областям, но рассмотренные выше методы имеют более широкую область применения. Дело в том, что при помощи подходящей замены переменных, некоторые регулярные области могут быть преобразованы в прямоугольные области. В частности это касается таких областей как круг, сфера, эллипс, эллипсоид и цилиндр (более подробная информация представлена в Приложении В). Например, чтобы вычислить интеграл по кругу, перейдём в полярные координаты и после преобразования получим

$$\iint_{x^2+y^2 \leq R^2} f(x,y) dx dy = \int_0^R \int_0^{2\pi} f(r \cos \varphi, r \sin \varphi) dr d\varphi .$$

Таким образом, исходный интеграл превращается в интеграл по прямоугольнику $0 \leq r \leq R$, $0 \leq \varphi \leq 2\pi$.

Более серьёзная проблема возникает, если нам нужно вычислить интеграл по некоторой нерегулярной области. Предположим, дана область S , показанная на рисунке 5.6, и нам необходимо приближённо вычислить

$$\iint_S f(x,y) dx dy .$$

При этом появляется дополнительная задача: как представить область S ? Наиболее простой способ - приблизить границу S ломаной линией, а затем представить область S как набор треугольников (смотри рисунок 5.6). Эта процедура называется триангуляцией. Теперь для того чтобы получить значение интеграла по области S , мы должны вычислить объёмы всех прямых треугольных призм, где высоты этих призм определяются функцией $f(x,y)$.

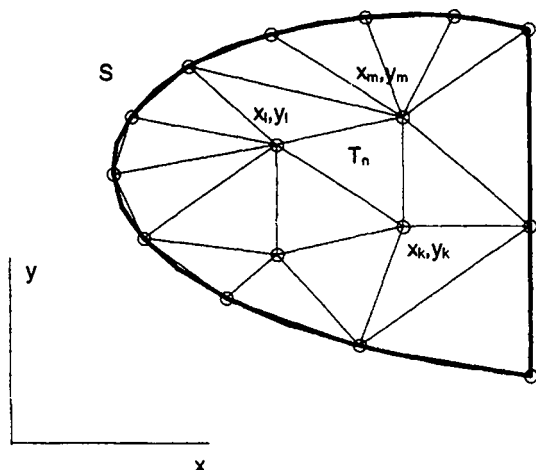


Рисунок 5.6. Триангуляция нерегулярной области

Прежде чем выводить формулу интегрирования по области S , мы построим кубатурную формулу для стандартного треугольника T с вершинами $(0, 0)$, $(0, 1)$ и $(1, 0)$ (рисунок 5.7). Если применить рассмотренный выше подход (смотри выражения (5.20) и (5.21)), то можно получить следующие формулы:

$$\iint_T f(r, s) dr ds = \frac{1}{6} (f(\beta_1, \gamma_1) + f(\beta_2, \gamma_2) + f(\beta_3, \gamma_3))$$

где

$$\beta_1 = 0, \beta_2 = 0.5, \beta_3 = 0.5, \gamma_1 = 0.5, \gamma_2 = 0.5, \gamma_3 = 0$$

или

$$\beta_1 = 1/6, \beta_2 = 2/3, \beta_3 = 1/6, \gamma_1 = 2/3, \gamma_2 = 1/6, \gamma_3 = 1/6$$

Можно легко проверить, что приведённая выше формула точна для одночленов вида $1, x, y, x^2, xy, y^2$. Трансформируя стандартный треугольник, можно получить формулу интегрирования по произвольному треугольнику T_n с вершинами

$$(x_k^{(n)}, y_k^{(n)}), (x_l^{(n)}, y_l^{(n)}) \text{ и } (x_m^{(n)}, y_m^{(n)}).$$

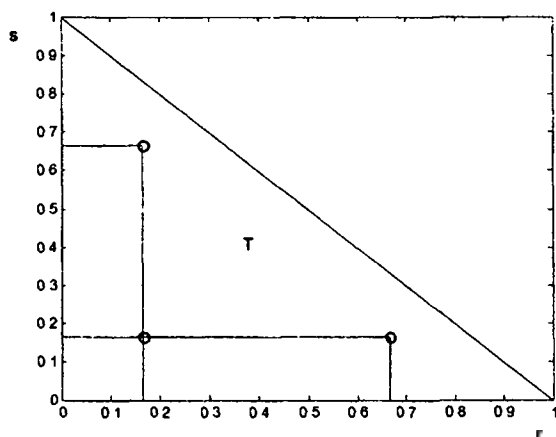
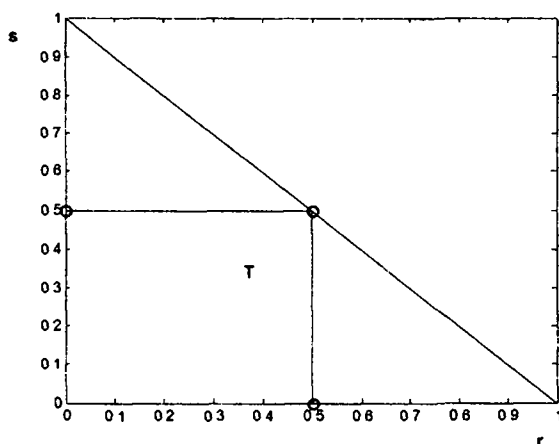


Рисунок 5.7 Узлы интегрирования для стандартного треугольника.

Суммируя значения интегралов по всем таким треугольникам, мы придём к общей формуле интегрирования по произвольной области S :

$$I \approx \frac{1}{3} \sum_{n=1}^N S_n (f(\bar{x}_1^{(n)}, \bar{y}_1^{(n)}) + f(\bar{x}_2^{(n)}, \bar{y}_2^{(n)}) + f(\bar{x}_3^{(n)}, \bar{y}_3^{(n)})),$$

где

$$\bar{x}_j^{(n)} = (x_m^{(n)} - x_k^{(n)})\beta_j + (x_l^{(n)} - x_k^{(n)})\gamma_j + x_k^{(n)},$$

$$\bar{y}_j^{(n)} = (y_m^{(n)} - y_k^{(n)})\beta_j + (y_l^{(n)} - y_k^{(n)})\gamma_j + y_k^{(n)},$$

$$j = 1, 2, 3$$

$$S_n - \text{площадь треугольника } T_n: S_n = \sqrt{p(p-a)(p-b)(p-c)},$$

$$a = \sqrt{(x_l^{(n)} - x_k^{(n)})^2 + (y_l^{(n)} - y_k^{(n)})^2}$$

$$b = \sqrt{(x_l^{(n)} - x_m^{(n)})^2 + (y_l^{(n)} - y_m^{(n)})^2}$$

$$c = \sqrt{(x_m^{(n)} - x_k^{(n)})^2 + (y_m^{(n)} - y_k^{(n)})^2}$$

$$p = \frac{1}{2}(a+b+c),$$

N – число треугольников приближающих область S .

Аналогичный подход может быть применен для численного интегрирования по нерегулярной трёхмерной области. В этом случае такая область представляется как набор тетраэдров.

ГЛАВА 6

Введение в конечно-разностные схемы для обыкновенных дифференциальных уравнений

Список обозначений

$u(x)$	точное решение дифференциального уравнения
$u_n^{(e)} = u(x_n)$	точное решение дифференциального уравнения в точке x_n
$u_n^{(a)}$	приближённое решение дифференциального уравнения в точке x_n
$e_n = u_n^{(e)} - u_n^{(a)}$	абсолютная ошибка приближённого решения в точке x_n
h	шаг разностной сетки
Du	дифференциальный оператор
$D_h u$	конечно-разностный оператор
$\delta f^{(a)}$	невязка
R	оператор перехода
$F(x, u) = \left\{ \frac{\partial f_n}{\partial x_m}(x, u) \right\}$	матрица Якоби
$\lambda_m(A)$	m -ое собственное значение матрицы A
ϵ_p	требуемая точность вычисления

Математические модели различных физических явлений часто формулируются в виде дифференциальных уравнений, и изучение таких моделей требует решения этих уравнений. В некоторых случаях их решение может быть выражено через известные функции. Однако, как правило, это невозможно, поэтому получение решения в виде явных выражений нельзя рассматривать как стандартный метод решения дифференциальных уравнений. Нельзя сказать, что аналитические методы

потеряли своё значение. Они остаются необходимым инструментом для решения упрощённых, так называемых, модельных задач. Исследование модельных задач позволяет получить важную информацию о решении более сложных задач.

Наряду с аналитическими методами широко используются различные численные методы для решения дифференциальных уравнений. Мы рассмотрим методы, основанные на конечных разностях. Основная идея этих методов состоит в том, что приближённое решение определяется на некотором множестве точек обычно называемом сеткой. Для вычисления этого приближенного решения используются алгебраические уравнения, которые приближают и заменяют дифференциальное уравнение.

6.1 Простейший пример конечно-разностной схемы

В качестве вводного примера рассмотрим построение вычислительной схемы для решения уравнения

$$\frac{du}{dx} + \frac{x}{1+x} u = 0, 0 \leq x \leq 1 \quad (6.1)$$

$$u(0) = 1$$

Первым шагом мы введём некоторое множество точек (узлов) на интервале $[0,1]$: $0 = x_0 < x_1 < \dots < x_N = 1$. Это простейший пример вычислительной сетки, и расстояние между двумя соседними узлами $h_n = x_{n+1} - x_n$ называется шагом сетки. В дальнейшем мы будем рассматривать более простой случай равномерной сетки, когда $h_n = h = \text{const}$. Приближённое решение уравнения (6.1) определяется в узлах сетки, то есть оно представляет собой множество значений

$$u_n^{(a)} = u^{(a)}(x_n), n = 0, \dots, N.$$

Такое множество часто называют сеточной функцией. Теперь нам нужно заменить производную некоторым разностным отношением, определённым на сетке. Простейший способ такой замены основан на определении производной:

$$\frac{du}{dx}(x) \approx \frac{u(x + \Delta x) - u(x)}{\Delta x},$$

где Δx выбирается достаточно малым. Применяя это приближение, мы получим, вместо дифференциального уравнения (6.1), разностное уравнение в каждом узле сетки ($x \rightarrow x_n$, $\Delta x \rightarrow h$)

$$\frac{u_{n+1}^{(a)} - u_n^{(a)}}{h} + \frac{x_n}{1 + x_n} u_n^{(a)} = 0,$$

$$u_0^{(a)} = 1, \quad (6.2)$$

$$x_n = nh, \quad n = 0, \dots, N-1, \quad h = \frac{1}{N}$$

Набор разностных уравнений определённых на некоторой сетке называется конечно-разностной схемой (или просто разностной схемой). Для того чтобы вычислить приближённое решение мы перепишем разностную схему (6.2) в форме рекуррентного соотношения

$$u_{n+1}^{(a)} = \left(1 - \frac{hx_n}{1 + x_n} \right) u_n^{(a)}.$$

Начиная с

$$u_0^{(a)} = 1,$$

для последовательных значений $n=0, \dots, N-1$ мы найдём приближённое решение во всех узлах сетки. Вообще говоря, можно построить бесконечное количество разностных схем для данного дифференциального уравнения. Например, можно записать следующие разностные схемы для уравнения (6.1):

$$\frac{u_{n+1}^{(a)} - u_n^{(a)}}{h} + \frac{x_{n+1}}{1 + x_{n+1}} u_{n+1}^{(a)} = 0, \quad (6.3)$$

с рекуррентным соотношением

$$u_{n+1}^{(a)} = \left(\frac{1 + x_{n+1}}{1 + (1+h)x_{n+1}} \right) u_n^{(a)}$$

или

$$\frac{u_{n+1}^{(a)} - u_{n-1}^{(a)}}{2h} + \frac{x_n}{1 + x_n} u_n^{(a)} = 0, \quad (6.4)$$

с рекуррентным соотношением

$$u_{n+1}^{(a)} = u_{n-1}^{(a)} - \frac{2hx_n}{1+x_n} u_n^{(a)}.$$

Естественно, что все эти схемы имеют разные свойства. Эти различия показаны на рисунке 6.1, который представляет результаты вычислений для разностных схем (6.2), (6.3) и (6.4), вместе с точным решением уравнения (6.1).

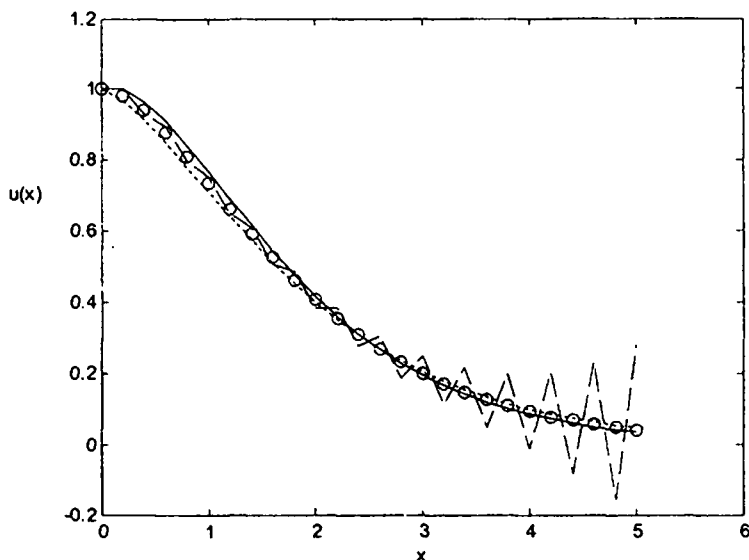


Рисунок 6.1 Приближённые решения задачи (6.1) полученные с помощью различных разностных схем ($l = 5$, $h = 0.2$); о - точное решение $u(x) = (1+x)\exp(-x)$; — - схема (6.2); --- - схема (6.3); — — - схема (6.4).

Этот пример демонстрирует, что построение разностных схем и решение дифференциальных уравнений с помощью этих схем не совсем простая задача. Часто возникают ситуации когда, казалось бы, схема должна давать достоверный результат, а мы получаем приближённое решение, которое не имеет ничего общего с точным решением. Поэтому, прежде чем рассматривать методы построения разностных

схем, мы должны познакомиться с требованиями, выполнение которых обязательно для любой практически пригодной схемы.

6.2 Определения аппроксимации и устойчивости.

В этом параграфе мы рассмотрим определения аппроксимации, устойчивости и сходимости разностной схемы. Предположим, что дифференциальная задача определена на интервале I . Это значит что требуется найти решение дифференциального уравнения (или системы уравнений) $u(x)$ на интервале I при дополнительных начальных и/или граничных условиях налагаемых на $u(x)$. Далее мы будем записывать дифференциальную задачу в символической форме:

$$Du = f, \quad (6.5)$$

где D - дифференциальный оператор и f - правая часть. Например, чтобы записать задачу (6.1) в форме (6.5) мы определим

$$Du \equiv \begin{cases} \frac{du}{dx} + \frac{x}{1+x}u, & 0 \leq x \leq 1 \\ u(0) \end{cases}, \quad f \equiv \begin{cases} 0 \\ 1 \end{cases}.$$

Другой пример: краевая задача

$$\frac{d^2 u}{dx^2} + a(x)u = g(x), \quad 0 \leq x \leq 1$$

$$u(0) = 1$$

$$\frac{du}{dx}(1) = 0$$

может быть представлена в виде (6.5), если записать

$$Du \equiv \begin{cases} \frac{d^2 u}{dx^2} + a(x)u, & 0 \leq x \leq 1 \\ u(0) \\ \frac{du}{dx}(1) \end{cases}, \quad f \equiv \begin{cases} g(x) \\ 1 \\ 0 \end{cases}.$$

6.2.1 Аппроксимация дифференциального уравнения разностной схемой.

Предположим что решение $u(x)$ задачи (6.5) существует на интервале I . Для того чтобы вычислить это решение методом конечных разностей введём конечное число точек (узлов) на интервале I . Это множество точек называется сеткой, и будет обозначаться как $I_h = \{x_n\}$ (рисунок 6.2).

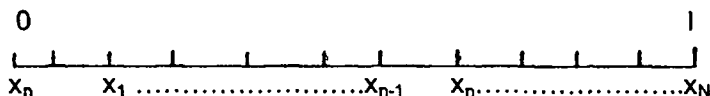


Рисунок 6.2. Одномерная сетка

Теперь мы будем искать не решение $u(x)$ задачи (6.5), а сеточную функцию

$$u^{(e)} = (u_1^{(e)}, \dots, u_N^{(e)}),$$

которая представляет собой множество значений решения в узлах I_h :

$$u_n^{(e)} = u(x_n).$$

Сетка I_h зависит от положительного параметра $h = x_{n+1} - x_n$, который может быть выбран сколь угодно малым. По мере увеличения количества узлов сетки, $h \rightarrow 0$, и сеточная функция $u^{(e)}$ будет давать всё более полное представление решения. Тогда используя интерполяцию можно получить решение в любой точке интервала I . Однако точно вычислить $u^{(e)}$ невозможно. Вместо сеточной функции $u^{(e)}$, мы можем вычислить другую сеточную функцию

$$u^{(a)} = (u_1^{(a)}, \dots, u_N^{(a)}),$$

которая стремится к $u^{(e)}$, когда число узлов сетки увеличивается. Для вычисления этого приближённого решения $u^{(a)}$ используются разностные схемы. По аналогии с выражением (6.5), разностную схему можно записать в символическом виде

$$D_h u^{(a)} = f^{(a)} \quad (6.6)$$

Например, разностная схема (6.2) может быть представлена в виде (6.6) если записать

$$D_h u^{(a)} \equiv \begin{cases} \frac{u_{n+1}^{(a)} - u_n^{(a)}}{h} + \frac{x_n}{1+x_n} u_n^{(a)}, & n = 0, \dots, N-1 \\ u_0^{(a)} \end{cases},$$

$$f^{(a)} \equiv \begin{cases} 0, & n = 0, \dots, N-1 \\ 1 \end{cases}.$$

Прежде чем продолжить наше обсуждение нам нужно определить способ измерения сеточных функций. На самом деле, этот способ нам уже известен, так как функции определённые на сетке h есть элементы линейного пространства R^N (вектора). Поэтому все определения, рассмотренные в параграфе 2.1, здесь применимы и величина отклонения $u^{(a)}$ от $u^{(e)}$ определяется как $\|u^{(e)} - u^{(a)}\|$.

Если подставить $u^{(e)}$ в разностную схему (6.6), то мы не получим точного равенства, то есть возникает некоторый вектор невязки:

$$D_h u^{(e)} = f^{(a)} + \delta f^{(a)}.$$

Поясним это на примере разностной схемы (6.2) для задачи (6.1). Точное решение задачи (6.1) имеет вид $u(x) = (1+x)\exp(-x)$. Если мы подставим сеточную функцию

$$u^{(e)} = \{u_n^{(e)} = (1+x_n)\exp(-x_n)\}$$

в разностную схему (6.2), то получим следующее выражение:

$$D_h u^{(e)} = \begin{cases} \frac{u_{n+1}^{(e)} - u_n^{(e)}}{h} + \frac{x_n}{1+x_n} u_n^{(e)} = \\ u_0^{(e)} \end{cases} = \begin{cases} \exp(-x_n) \left[\frac{(1+x_n)(\exp(-h)-1) + h \exp(-h)}{h} + x_n \right], \\ n = 0, \dots, N-1 \\ 1 \end{cases}$$

Когда $h \rightarrow 0$,

$$\exp(-h) = 1 - h + \frac{1}{2}h^2 + O(h^3)$$

и предыдущее выражение принимает вид

$$D_h u^{(e)} = f^{(a)} + \begin{cases} -\frac{1}{2} h \exp(-x_n)(1-x_n) + O(h^2), \\ n = 0, \dots, N-1 \\ 0 \end{cases} = f^{(a)} + \delta f^{(a)}$$

Мы будем говорить, что разностная схема

$$D_h u^{(a)} = f^{(a)}$$

аппроксимирует дифференциальную задачу $Du=f$ на решении $u(x)$ если $\|\delta f^{(a)}\| \rightarrow 0$ когда $h \rightarrow 0$. В дополнение, если выполняется неравенство $\|\delta f^{(a)}\| \leq Ch^k$,

где $C > 0$ и $k > 0$, тогда мы будем говорить, что имеет место аппроксимация k -го порядка или, разностная схема имеет порядок k .

Так как точное решение исходной задачи нам неизвестно, мы можем сделать только оценку невязки. Далее мы рассмотрим пример, который демонстрирует общепринятый метод исследования аппроксимации.

Пример 6.1 (анализ аппроксимации)

Рассмотрим дифференциальную задачу (6.1). В параграфе 6.1 приведены три разностные схемы для этой задачи, и наша цель исследовать аппроксимацию этих схем. Вначале рассмотрим разностную схему (6.2). Если вместо сеточной функции $u^{(a)}$ подставить точное решение $u(x)$, то выражение (6.2) примет вид

$$D_h u^{(e)} = \begin{cases} \frac{u(x_{n+1}) - u(x_n)}{h} + \frac{x_n}{1+x_n} u(x_n), \\ n = 0, \dots, N-1 \\ u(0) \end{cases} = \begin{cases} 0 \\ 1 \end{cases} \quad (6.7)$$

Предположим что все производные функции $u(x)$, до четвёртого порядка включительно, непрерывны и ограничены. Тогда мы можем

разложить $u(x_{n+1})=u(x_n+h)$ по степеням h в окрестности точки x_n по формуле Тейлора:

$$u(x_{n+1}) = u(x_n + h) = u(x_n) + h \frac{du}{dx}(x_n) + \frac{1}{2} h^2 \frac{d^2 u}{dx^2}(x_n) + O(h^3)$$

Подставляя это разложение в (6.7), получим

$$\frac{u(x_{n+1}) - u(x_n)}{h} + \frac{x_n}{1+x_n} u(x_n) = \left[\frac{du}{dx}(x_n) + \frac{x_n}{1+x_n} u(x_n) \right] + \frac{1}{2} h \frac{d^2 u}{dx^2}(x_n) + O(h^2),$$

$$n = 1, \dots, N-1$$

Выражение в скобках есть левая часть уравнения (6.1) вычисленная в узлах сетки и, следовательно, оно равно нулю. Тогда выражение (6.7) можно записать как

$$D_h u^{(e)} = f^{(a)} + \delta f^{(a)} = \begin{cases} 0 \\ 1 \end{cases} + \begin{cases} \frac{1}{2} h \frac{d^2 u}{dx^2}(x_n) + O(h^2), n = 1, \dots, N-1 \\ 0 \end{cases}$$

Когда шаг разностной сетки достаточно мал, компоненты вектора невязки пропорциональны h и $\|\delta f^{(a)}\| \leq Ch$. Следовательно, разностная схема (6.2) аппроксимирует дифференциальную задачу (6.1) с первым порядком по h .

Теперь обратимся к разностной схеме (6.3). После замены $u^{(a)}$ на точное решение, мы получим

$$D_h u^{(e)} = \begin{cases} \frac{u(x_{n+1}) - u(x_n)}{h} + \frac{x_{n+1}}{1+x_{n+1}} u(x_{n+1}), \\ n = 0, \dots, N-1 \\ u(0) \end{cases} = \begin{cases} 0 \\ 1 \end{cases} \quad (6.8)$$

В этом случае, анализ удобнее проводить в точке x_{n+1} и разложение $u(x_n)=u(x_{n+1}-h)$ по степеням h в окрестности x_{n+1} даёт

$$u(x_n) = u(x_{n+1}) - h \frac{du}{dx}(x_{n+1}) + \frac{1}{2} h^2 \frac{d^2 u}{dx^2}(x_{n+1}) + O(h^3)$$

Подставляя это разложение в (6.8), получим

$$\begin{aligned} \frac{u(x_{n+1}) - u(x_n)}{h} + \frac{x_{n+1}}{1+x_{n+1}} u(x_{n+1}) = \\ \left[\frac{du}{dx}(x_{n+1}) + \frac{x_{n+1}}{1+x_{n+1}} u(x_{n+1}) \right] - \frac{1}{2} h \frac{d^2 u}{dx^2}(x_{n+1}) + O(h^2). \end{aligned}$$

$$n = 0, \dots, N-1$$

Тогда мы можем заключить, что разностная схема (6.2) аппроксимирует дифференциальную задачу (6.1) с первым порядком по h .

В заключение рассмотрим разностную схему (6.4), которая на решении $u(x)$ имеет следующий вид

$$D_n u^{(e)} = \begin{cases} \frac{u(x_{n+1}) - u(x_{n-1}))}{2h} + \frac{x_n}{1+x_n} u(x_n), \\ n = 0, \dots, N-1 \\ u(0) \end{cases} = \begin{cases} 0 \\ 1 \end{cases} \quad (6.9)$$

Исследовать аппроксимацию будем в точке x_n , поэтому нам нужно разложить $u(x_{n+1})=u(x_n+h)$ и $u(x_{n-1})=u(x_n-h)$ по степеням h в окрестности x_n :

$$\begin{aligned} u(x_n \pm h) = u(x_n) \pm h \frac{du}{dx}(x_n) + \frac{1}{2} h^2 \frac{d^2 u}{dx^2}(x_n) \pm \\ \frac{1}{6} h^3 \frac{d^3 u}{dx^3}(x_n) + O(h^4) \end{aligned}$$

Подставляя это выражение в (6.9), мы получим

$$\begin{aligned} \frac{u(x_{n+1}) - u(x_{n-1}))}{2h} + \frac{x_n}{1+x_n} u(x_n) = \\ \left[\frac{du}{dx}(x_n) + \frac{x_n}{1+x_n} u(x_n) \right] + \frac{1}{6} h^2 \frac{d^3 u}{dx^3}(x_n) + O(h^3). \end{aligned}$$

$$n = 1, \dots, N-1$$

Теперь мы имеем $\|\delta f^{(a)}\| \leq Ch^2$ и это говорит о том, что разностная схема (6.4) является схемой второго порядка.

6.2.2 Замена производных разностными отношениями

В рассмотренных выше примерах мы строили разностные схемы, заменяя производные в дифференциальных уравнениях на разностные отношения. Это общий подход, который позволяет конструировать разностные схемы любого порядка аппроксимации для дифференциальных задач с достаточно гладким решением $u(x)$. Действительно, можно заменить производную du^k/dx^k таким разностным выражением, что ошибка, возникающая при такой замене, будет иметь любой заданный порядок p относительно h для достаточно гладкой функции $u(x)$. Рассмотрим процедуру построения разностных отношений основанную на методе неопределённых коэффициентов. Общая формула для аппроксимации производных в некоторой точке x_n имеет следующий вид:

$$\frac{d^k u}{dx^k}(x_n) = \frac{1}{h^k} \sum_{s=-m}^l a_s u(x_n + sh) + O(h^p) \quad (6.10)$$

где коэффициенты a_s ($s=-m, -m+1, \dots, l$) подбираются таким образом, чтобы добиться требуемого порядка аппроксимации. Пределы суммирования, m и l , подчиняются условию $m+l \geq k+p-1$. Из формулы Тейлора следует, что

$$u(x_n + sh) = u(x_n) + sh \frac{du}{dx}(x_n) + \frac{1}{2!} (sh)^2 \frac{d^2 u}{dx^2}(x_n) + \dots +$$

$$\frac{1}{(k+p-1)!} (sh)^{k+p-1} \frac{d^{k+p-1} u}{dx^{k+p-1}}(x_n) + O(h^{k+p})$$

Подставим это разложение вместо $u(x_n+sh)$ в выражении (6.10) и получим

$$\frac{d^k u}{dx^k}(x_n) = \frac{1}{h^k} \left[u(x_n) \sum_{s=-m}^l a_s + h \frac{du}{dx}(x_n) \sum_{s=-m}^l s a_s + \dots + \right.$$

$$\left. \frac{h^{k+p-1}}{(k+p-1)!} \frac{d^{k+p-1} u}{dx^{k+p-1}}(x_n) \sum_{s=-m}^l s^{k+p-1} a_s \right] + O(h^p)$$

Приравнявая коэффициенты при h^s , $s = -k, -k+1, \dots, p-1$ в левой и правой частях этого выражения, мы придём к следующей системе уравнений для a_s :

$$\begin{aligned} \sum_{s=-m}^l a_s &= 0, \\ \sum_{s=-m}^l s a_s &= 0, \\ &\dots\dots\dots, \\ \sum_{s=-m}^l s^{k-1} a_s &= 0, \\ \sum_{s=-m}^l s^k a_s &= k!, \\ \sum_{s=-m}^l s^{k+1} a_s &= 0, \\ &\dots\dots\dots, \\ \sum_{s=-m}^l s^{k+p-1} a_s &= 0. \end{aligned} \tag{6.11}$$

Если $m+l=k+p-1$, то мы имеем систему из $k+p$ линейных уравнений для $k+p$ неизвестных a_s . Определитель матрицы коэффициентов этой системы есть определитель Вандермонда, который не равен нулю. Таким образом, в этом случае существует единственный набор коэффициентов a_s , удовлетворяющий системе (6.11). Если $m+l \geq k+p$, то решение системы (6.11) не единственно.

Теперь рассмотрим примеры разностных выражений, которые часто используются на практике. Существует единственное выражение вида

$$\frac{1}{h} [a_0 u(x_n) + a_1 u(x_n + h)],$$

аппроксимирующее du/dx с первым порядком по h . Система (6.11) в этом случае имеет простой вид

$$\begin{aligned} a_0 + a_1 &= 0 \\ a_1 &= 1 \end{aligned},$$

с решением $a_0 = -1$, $a_1 = 1$. Следовательно, мы получаем

$$\frac{du}{dx}(x_n) = \frac{u(x_n + h) - u(x_n)}{h} + O(h),$$

и это соотношение часто называют «разностью вперёд».

Аналогично можно показать, что существует единственное выражение вида

$$\frac{1}{h} [a_{-1}u(x_n - h) + a_0u(x_n)]$$

аппроксимирующее du/dx с первым порядком по h , то есть

$$\frac{du}{dx}(x_n) = \frac{u(x_n) - u(x_n - h)}{h} + O(h).$$

Это соотношение часто называют «разностью назад».

Если мы хотим аппроксимировать du/dx со вторым порядком по h , тогда мы имеем $m+1 = 2$ и выражение (6.10) принимает вид

$$\frac{1}{h} [a_{-1}u(x_n - h) + a_0u(x_n) + a_1u(x_n + h)].$$

Система уравнений (6.11) в этом случае записывается как

$$\begin{aligned} a_{-1} + a_0 + a_1 &= 0 \\ a_1 - a_{-1} &= 1 \\ a_1 + a_{-1} &= 0 \end{aligned}$$

Решая эту систему, получим, что $a_{-1} = -1/2$, $a_0 = 0$, $a_1 = 1/2$ и, следовательно

$$\frac{du}{dx}(x_n) = \frac{u(x_n + h) - u(x_n - h)}{2h} + O(h^2).$$

Это соотношение часто называют «центральной разностью».

Рассмотрим теперь аппроксимацию производной d^2u/dx^2 с порядком h^2 . В данном случае, $k=2$, $p=2$ и при условии $m+1=3$ выражение вида

$$\frac{1}{h^2} [a_{-1}u(x_n - h) + a_0u(x_n) + a_1u(x_n + h) + a_2u(x_n + 2h)]$$

является единственным выражением с необходимым нам свойством. Решая соответствующую систему (6.11) для коэффициентов $a_{-1}, \dots, a_2$, мы найдём, что $a_{-1} = a_1 = 1$, $a_0 = -2$, $a_2 = 0$ и, окончательно, мы приходим к следующему разностному отношению

$$\frac{d^2u}{dx^2}(x_n) = \frac{u(x_n + h) - 2u(x_n) + u(x_n - h)}{h^2} + O(h^2).$$

6.2.3 Определенне устойчивости разностной схемы

Как было показано в параграфе 6.1, приближённое решение $u^{(a)}$, полученное с помощью разностной схемы (6.4), не отображает точного решения, хотя эта схема аппроксимирует задачу (6.1) со вторым порядком по h . Разностная схема должна обладать не только свойством аппроксимации, но также быть устойчивой.

Рассмотрим возмущённую разностную схему

$$D_h z^{(a)} = f^{(a)} + \epsilon^{(a)}, \quad (6.12)$$

которая получается из разностной схемы (6.6) путём добавления в правую часть возмущения $\epsilon^{(a)}$. Мы будем называть разностную схему (6.6) устойчивой, если существуют числа $h_0 > 0$ и $\delta > 0$, такие что для любого $h < h_0$ и $\|\epsilon^{(a)}\| < \delta$ разностная задача (6.12) имеет единственное решение $z^{(a)}$ и это решение отличается от $u^{(a)}$ на сеточную функцию $z^{(a)} - u^{(a)}$, которая удовлетворяет оценке $\|z^{(a)} - u^{(a)}\| \leq C \|\epsilon^{(a)}\|$, где константа C не зависит от h . Это неравенство означает, что малое возмущение правой части разностной схемы (6.6) приводит к малому возмущению решения.

В случае линейной дифференциальной задачи (6.5), следующее определение эквивалентно данному выше определению. Мы будем называть разностную схему (6.6) устойчивой, если для любой правой части $f^{(a)}$ разностная задача $D_h u^{(a)} = f^{(a)}$ имеет единственное решение $u^{(a)}$ и выполняется оценка

$$\|u^{(a)}\| \leq C \|f^{(a)}\|, \quad (6.13)$$

где константа C не зависит от h .

6.2.4 Сходимость как следствие аппроксимации и устойчивости.

Основной вопрос, который возникает при практических вычислениях - насколько полученное приближённое решение отличается от точного решения и как сделать это отклонение достаточно малым.

Предположим что разностная схема (6.6) аппроксимирует задачу (6.5) на решении $u(x)$ с порядком h^k и устойчива. Тогда приближённое решение $u^{(a)}$ сходится к точному решению $u^{(e)}$, то есть $\|u^{(e)} - u^{(a)}\| \rightarrow 0$ когда $h \rightarrow 0$. В дополнение выполняется неравенство

$$\|u^{(e)} - u^{(a)}\| \leq Ch^k$$

где константа C не зависит от h . В этом случае мы будем говорить, что имеет место сходимость порядка h^k или, что разностная схема (6.6) имеет точность k -го порядка. Понятно, что чем больше значение k (выше порядок аппроксимации), тем быстрее приближённое решение стремится к точному решению по мере уменьшения шага сетки. Наличие сходимости является основным требованием, предъявляемым к разностным схемам, так как только в этом случае мы можем аппроксимировать точное решение $u^{(e)}$ с любой точностью, в пределах машинной арифметики, выбирая шаг h достаточно малым.

Пример 6.2 (сходимость разностной схемы первого порядка)

Рассмотрим разностную схему (6.2). Как уже было показано выше, эта схема аппроксимирует дифференциальную задачу (6.1) с порядком h и устойчива, по крайней мере, при $h \leq 0.2$. Продемонстрируем сходимость этой схемы, так как точное решение дифференциальной задачи (6.1) известно. Для вычисления $\|u^{(e)} - u^{(a)}\|$ мы будем использовать норму $\|\cdot\|_\infty$, то есть

$$\|u^{(e)} - u^{(a)}\|_\infty = \max_{0 \leq n \leq N} |u_n^{(e)} - u_n^{(a)}|.$$

Результаты вычислений показаны в Таблице 6.1.

Таблица 6.1

h	$\ u^{(e)} - u^{(a)}\ _\infty / \ u^{(e)}\ _\infty$
0.2	0.0342
0.1	0.0161
0.05	0.0078
0.025	0.0038
0.0125	0.0019

Эти результаты показывают, что схема первого порядка имеет довольно медленную сходимость. Например, если мы хотим вычислить приближённое решение с относительной точностью меньше чем 0.001, то мы должны выбрать шаг меньше чем 0.01. Конечно, в реальной ситуации точное решение задачи нам не известно. Тем не менее, можно оценить сходимость через приближённые решения. Более подробно мы обсудим этот вопрос в параграфе 6.5.

6.3 Численное решение задачи Коши.

Вначале этого параграфа мы рассмотрим численные методы решения дифференциальных задач вида

$$\frac{du}{dx} = f(x, u), \quad a \leq x \leq b \quad (6.14)$$

с начальным условием $u(a) = \alpha$.

Далее мы обсудим обобщение этих методов на случай систем дифференциальных уравнений первого порядка

$$\frac{du}{dx} = f(x, u), \quad a \leq x \leq b \quad (6.15)$$

или в развёрнутом виде

$$\begin{aligned} \frac{du_1}{dx} &= f_1(x, u_1, \dots, u_N), \\ &\dots\dots\dots, \\ \frac{du_N}{dx} &= f_N(x, u_1, \dots, u_N), \end{aligned}$$

с начальными условиями

$$\begin{aligned} u_1(a) &= \alpha_1, \\ &\dots\dots\dots \\ u_N(a) &= \alpha_N \end{aligned}$$

Мы предполагаем, что задача Коши (6.15) (или (6.14)) имеет единственное и устойчивое решение. Решение $u(x)$ задачи (6.15) называется

- устойчивым, если для любого $\epsilon > 0$ существует $\delta > 0$ такое, что из выполнения условия

$$\|u(a) - u^*(a)\| \leq \delta,$$

где $u^*(x)$ есть другое решение уравнения (6.15), следует выполнение условия

$$\|u(x) - u^*(x)\| \leq \epsilon$$

для всех $x \leq a$.

- асимптотически устойчивым, если, будучи устойчивым,

$$\|u(x) - u^*(x)\| \rightarrow 0$$

когда $x \rightarrow \infty$.

Решение $u(x)$ называется невозмущенным, а $u^*(x)$ возмущённым. Другими словами, устойчивость означает, что малые возмущения в начальных условиях приводят к малым возмущениям решения. В случае линейной задачи вида

$$\frac{du}{dx} = Au + g(x),$$

где A – постоянная матрица размером N на N , устойчивость можно определить через свойства этой матрицы. Решение приведённой выше задачи является

- устойчивым тогда и только тогда, когда все собственные значения матрицы A имеют неположительные вещественные части,
- асимптотически устойчивым тогда и только тогда, когда все собственные значения матрицы A имеют отрицательные вещественные части.

Для нелинейной задачи (6.15) можно провести локальный линейный анализ для того, чтобы проследить развитие малых возмущений. Если представить возмущённое решение в виде $u^*(x) = u(x) + v(x)$, где $u^*(a)$ близко к $u(a)$, то можно пренебречь слагаемым более высокой степени малости $g(x, u, v)$ в разложении

$$f(x, u^*) = f(x, u + v) = f(x, u) + F(x, u)v + g(x, u, v),$$

и рассматривать линейное уравнение

$$\frac{dv}{dx} = \frac{du^*}{dx} - \frac{du}{dx} = f(x, u^*) - f(x, u) = F(x, u)v$$

относительно v с матрицей Якоби

$$F(x, u) = \left\{ \frac{\partial f_n}{\partial x_m}(x, u) \right\}.$$

Мы рассмотрели лишь основные определения помогающие понять суть явления. Более полное освещение вопросов устойчивости нелинейных задач можно найти в учебниках по дифференциальным уравнениям.

Прежде чем описывать численные методы решения задачи (6.14) мы рассмотрим один из методов анализа устойчивости разностных схем для задачи Коши.

6.3.1 Необходимое условие устойчивости разностных схем для линейных задач.

При решении задачи Коши, компоненты сеточной функции $u^{(a)}$ вычисляются последовательно в узлах сетки. Если оценивать рост решения после каждого такого перехода, то мы получим некоторую процедуру исследования устойчивости. Вначале мы рассмотрим метод исследования устойчивости разностных схем для модельного уравнения

$$\begin{aligned} \frac{du}{dx} &= -\gamma u, \quad \gamma > 0, \quad x \geq 0 \\ u(0) &= \alpha \end{aligned} \quad (6.16)$$

Любая разностная схема для этого уравнения может быть записана в следующей канонической форме:

$$\begin{aligned} y_{n+1} &= R y_n, \quad n = 0, \dots, N-1 \\ y_0 &\text{ задано,} \end{aligned} \quad (6.17)$$

где y_n (в общем виде это вектор) зависит от $u^{(a)}$, и $R=R(h)$ называется оператором перехода (в общем виде это матрица). В этом случае

$$\|u^{(a)}\| = \max_n \|y_n\|,$$

$$\|f^{(a)}\| = \|y_0\|.$$

Легко получить, что $y_n = R^n y_0$ и тогда мы можем записать

$$\begin{aligned} \|u^{(a)}\| &= \max_n \|y_n\| = \max_n \|R^n y_0\| \leq \\ &\max_n \|R^n\| \cdot \|y_0\| \leq \max_n \|R^n\| \cdot \|f^{(a)}\|. \end{aligned}$$

Согласно определению устойчивости (6.13) мы приходим к следующему условию

$$\max_n \|R^n\| \leq \text{const}$$

или

$$\|R^n\| \leq \|R\|^n \leq \text{const}, \quad n = 0, \dots, N-1,$$

которое означает ограниченность нормы степеней оператора перехода. Определённо это условие будет выполнено для любого значения N , если $\|R\| \leq 1$. Из линейной алгебры известно, что спектральный радиус матрицы меньше (или равен) любой нормы матрицы. Принимая это во внимание, мы получим необходимое условие устойчивости разностной схемы для задачи (6.16):

$$s(\mathbf{R}) = \max_m |\lambda_m(\mathbf{R})| \leq 1, \quad (6.18)$$

где $\lambda_m(\mathbf{R})$ - собственные значения оператора перехода $\mathbf{R}$.

Пример 6.3 (анализ устойчивости)

В этом примере мы обратимся к анализу некоторых разностных схем для задачи (6.16). Вначале мы построим разностную схему аналогичную схеме (6.2):

$$\frac{u_{n+1}^{(a)} - u_n^{(a)}}{h} + \gamma u_n^{(a)} = 0, \quad n = 0, \dots, N-1$$

$$u_0^{(a)} = \alpha \quad (6.19)$$

Если мы перепишем эту схему в виде

$$u_{n+1}^{(a)} = (1 - \gamma h) u_n^{(a)}, \quad n = 0, \dots, N-1$$

$$u_0^{(a)} = \alpha$$

то получим каноническую форму (6.17), где $y_n = u_n$, $R = 1 - \gamma h$. Тогда критерий устойчивости (6.18) превращается в условие $|1 - \gamma h| \leq 1$. Разрешая это неравенство относительно h , мы получим следующее ограничение: $h \leq 2/\gamma$. Разностная схема (6.19) является примером условно устойчивой схемы. Такие схемы устойчивы только при выполнении определённых условий на шаг разностной сетки.

Теперь рассмотрим разностную схему аналогичную схеме (6.3):

$$\frac{u_{n+1}^{(a)} - u_n^{(a)}}{h} + \gamma u_{n+1}^{(a)} = 0, \quad n = 0, \dots, N-1$$

$$u_0^{(a)} = \alpha \quad (6.20)$$

Выражая из этого уравнения $u_{n+1}^{(a)}$, мы приходим к следующей канонической форме

$$u_{n+1}^{(a)} = \frac{u_n^{(a)}}{(1 + \gamma h)} = R u_n^{(a)}, \quad n = 0, \dots, N-1$$

$$u_0^{(a)} = \alpha$$

Критерий устойчивости (6.18) теперь принимает вид

$$\frac{1}{(1+\gamma h)} \leq 1,$$

и он выполняется для любого $h>0$. Разностная схема (6.20) является примером безусловно устойчивой схемы.

В заключение мы рассмотрим разностную схему

$$\begin{aligned} \frac{u_{n+1}^{(a)} - u_{n-1}^{(a)}}{2h} + \gamma u_n^{(a)} &= 0, \quad n = 1, \dots, N-1 \\ u_0^{(a)} &= \alpha, \\ u_1^{(a)} &= (1 - \gamma h)\alpha \end{aligned} \quad (6.21)$$

которая аппроксимирует задачу (6.16) со вторым порядком по h . Мы не можем сразу записать эту схему в форме (6.17), так как она связывает не два, а три последовательных значения:

$$u_{n+1}^{(a)}, u_n^{(a)}, u_{n-1}^{(a)}.$$

Для того чтобы преодолеть это затруднение запишем (6.21) в следующем виде:

$$\begin{aligned} u_{n+1}^{(a)} &= u_{n-1}^{(a)} - 2\gamma h u_n^{(a)}, \\ u_n^{(a)} &= u_n^{(a)} \end{aligned}$$

Если мы введём вектор

$$y_n = \begin{pmatrix} u_n^{(a)} \\ u_{n-1}^{(a)} \end{pmatrix},$$

тогда предыдущее выражение можно представить как

$$y_{n+1} = \begin{pmatrix} u_{n+1}^{(a)} \\ u_n^{(a)} \end{pmatrix} = \begin{pmatrix} -2\gamma h & 1 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} u_n^{(a)} \\ u_{n-1}^{(a)} \end{pmatrix} = R y_n$$

Теперь разностная схема (6.21) записана в форме (6.17). Собственные значения полученного оператора перехода есть корни полинома

$$\det \begin{pmatrix} -2\gamma h - \lambda & 1 \\ 1 & -\lambda \end{pmatrix} = \det(\mathbf{R} - \lambda \mathbf{I}) = 0,$$

и они имеют значения

$$\lambda_{1,2} = -\gamma h \pm \sqrt{1 + (\gamma h)^2}.$$

Это выражение показывает, что

$$|\lambda_2| = \gamma h + \sqrt{1 + (\gamma h)^2} > 1$$

для любого $h > 0$, поэтому разностная схема (6.21) неустойчива. Неустойчивые разностные схемы обычно дают осциллирующие решения возрастающей амплитуды, подобные тому, показанному на рисунке 6.1, и, естественно, такие решения не сходятся к точному решению когда $h \rightarrow 0$.

6.3.2 Методы Рунге-Кутты.

С простейшей схемой Рунге-Кутты (РК) для задачи (6.14), по сути дела, мы уже встречались (смотри схему (6.2)). Это схема Эйлера

$$\begin{aligned} \frac{u_{n+1}^{(a)} - u_n^{(a)}}{h} &= f(x_n, u_n^{(a)}), \\ u_0^{(a)} &= \alpha, \\ x_n &= a + nh, \quad n = 0, \dots, N-1, \quad h = \frac{b-a}{N} \end{aligned} \quad (6.22)$$

которая имеет первый порядок аппроксимации. Как мы видели в примере 6.2, схемы первого порядка сходятся довольно медленно к точному решению. В результате требуется значительный объем вычислений, чтобы достичь требуемой точности. В общем виде, явные РК схемы строятся следующим образом:

$$\frac{u_{n+1}^{(a)} - u_n^{(a)}}{h} = b_1 k_1 + \dots + b_s k_s, \quad n = 0, \dots, N-1, \quad (6.23)$$

$$u_0^{(a)} = \alpha,$$

$$k_1 = f(x_n, u_n^{(a)})$$

$$k_2 = f(x_n + c_2 h, u_n^{(a)} + a_{21} h k_1)$$

$$k_3 = f(x_n + c_3 h, u_n^{(a)} + h(a_{31} k_1 + a_{32} k_2))$$

.....

$$k_s = f(x_n + c_s h, u_n^{(a)} + h \sum_{m=1}^{s-1} a_{sm} k_m)$$

где s есть число стадий и b_i , c_i , $a_{i,m}$ некоторые вещественные параметры. Задавая число стадий, можно определить эти параметры таким образом, чтобы достичь наивысшего возможного порядка аппроксимации. Схемы Рунге-Кутты (6.23) часто представляются символически в виде таблицы Бучера (Butcher):

0	0				
c_2	$a_{2,1}$				
c_3	$a_{3,1}$	$a_{3,2}$			
.	.	.	.		
c_s	$a_{s,1}$	$a_{s,2}$	...	$a_{s,s-1}$	
	b_1	b_2	...	b_{s-1}	b_s

В качестве примера рассмотрим двухстадийную схему (6.23)

$$\frac{u_{n+1}^{(a)} - u_n^{(a)}}{h} = b_1 k_1 + b_2 k_2, \quad n = 0, \dots, N-1$$

$$u_0^{(a)} = \alpha,$$

$$k_1 = f(x_n, u_n^{(a)})$$

$$k_2 = f(x_n + c_2 h, u_n^{(a)} + a_{21} h k_1)$$

(6.24)

Для того чтобы определить параметры b_1 , b_2 , c_2 , и $a_{2,1}$, мы должны провести анализ аппроксимации. Если заменить $u^{(a)}$ в (6.24) на точное решение $u(x)$, то мы можем записать в точке x_n

$$\begin{aligned} \frac{u(x_{n+1}) - u(x_n)}{h} &= \frac{du}{dx}(x_n) + \frac{1}{2}h \frac{d^2u}{dx^2}(x_n) + O(h^2) = \\ &= (b_1 + b_2)f(x_n, u(x_n)) + b_2c_2h \frac{\partial f}{\partial x}(x_n, u(x_n)) + \\ &+ b_2a_{2,1}hf(x_n, u(x_n)) \frac{\partial f}{\partial u}(x_n, u(x_n)) + O(h^2) \end{aligned}$$

Учитывая, что

$$\frac{d^2u}{dx^2} = \frac{d}{dx} \left(\frac{du}{dx} \right) = \frac{\partial f}{\partial x} + \frac{\partial f}{\partial u} \frac{du}{dx} = \frac{\partial f}{\partial x} + f \frac{\partial f}{\partial u},$$

предыдущее выражение переписывается в виде

$$\begin{aligned} \frac{du}{dx}(x_n) - (b_1 + b_2)f(x_n, u(x_n)) &= \\ &= h \left(b_2c_2 - \frac{1}{2} \right) \frac{\partial f}{\partial x}(x_n, u(x_n)) + \\ &+ h \left(b_2a_{2,1} - \frac{1}{2} \right) f(x_n, u(x_n)) \frac{\partial f}{\partial u}(x_n, u(x_n)) + O(h^2) \end{aligned}$$

Для того чтобы получить второй порядок аппроксимации, следует положить

$$b_1 + b_2 = 1$$

$$b_2c_2 = \frac{1}{2}$$

$$b_2a_{2,1} = \frac{1}{2}$$

Теперь мы имеем четыре неизвестных, но три уравнения, поэтому один параметр можно выбрать произвольно. В результате мы получим семейство разностных схем второго порядка аппроксимации, например:

$$\begin{array}{c|cc} 0 & & \\ 1 & 1 & \\ \hline & 1/2 & 1/2 \end{array} \quad (6.25)$$

или

0	
2/3	2/3
	1/4 3/4

Такая ситуация характерна для РК схем: существует семейство разностных схем для каждого порядка аппроксимации. Схемы более высокого порядка могут быть построены аналогичным образом, но процедура такого построения довольно громоздка и, поэтому рассматриваться не будет. Приведём только несколько примеров. Так

0		
1/2	1/2	
3/4	0	3/4
	2/9	1/3 4/9

(6.26)

задаёт коэффициенты для схемы третьего порядка и таблицы

0			
1/2	1/2		
1/2	0	1/2	
1	0	0	1
	1/6	2/6	2/6 1/6

(6.27)

и

0			
1/3	1/3		
2/3	-1/3	1	
1	1	-1	1
	1/8	3/8	3/8 1/8

задают коэффициенты для схем четвёртого порядка. Явные схемы Рунге-Кутты условно устойчивы. Области устойчивости для некоторых схем, в случае модельного уравнения (6.16), представлены в Прило-

жении С. Эти результаты применимы не только для схем (6.25), (6.26) и (6.27), так как все явные РК схемы с p стадиями порядка p имеют одинаковую область устойчивости.

Чем выше порядок разностной схемы, тем больше вычислений требуется на один шаг. С другой стороны, чем выше порядок разностной схемы, тем быстрее сходимость к точному решению. Поэтому, общее количество операций для схемы более высокого порядка может быть меньше по сравнению со схемой более низкого порядка, как демонстрируется в следующем примере.

Пример 6.4 (сходимость некоторых схем Рунге-Кутты)

Рассмотрим следующую дифференциальную задачу

$$\frac{du}{dx} = -\frac{u^2}{1+x}, \quad 0 \leq x \leq 10$$

$$u(0) = 1$$

Точное решение этой задачи имеет вид $u(x) = (1 + \ln(1+x))^{-1}$. Вычислим относительную ошибку приближенного решения

$$e_r = \frac{\|u^{(n)} - u^{(e)}\|_{\infty}}{\|u^{(e)}\|_{\infty}}$$

для схем (6.22), (6.25), (6.26) и (6.27) при разных шагах h . Результаты вычислений показаны в Таблице 6.2.

Таблица 6.2

h	e_r для схемы (6.22)	e_r для схемы (6.25)	e_r для схемы (6.26)	e_r для схемы (6.27)
1	5.9e-01	9.1e-02	6.3e-02	1.6e-02
0.5	2.1e-01	3.2e-03	9.3e-03	9.3e-04
0.25	7.4e-02	1.2e-03	1.0e-03	3.8e-05
....				
0.03125	7.4e-03	2.9e-05		

Предположим, что нам нужно приближенное решение с точностью $e_r \leq 10^{-4}$. Схема четвертого порядка (6.27) даёт такое решение при $h=0.25$ (что требует 160 вычислений функции правой части), в то время как

схема второго порядка (6.25) даёт похожий результат только при $h=0.03125$ (640 вычислений функции правой части).

6.3.3 Методы Адамса.

Для того чтобы получить u_{n+1} при заданном u_n используя схему Рунге-Кутты, необходимо вычислить функцию $f(x, u)$ s раз в некоторых промежуточных точках. Эти вычисленные значения далее больше не используются. В схемах Адамса, для вычисления следующего значения u_{n+1} используется не только значение u_n , но и несколько предыдущих значений. В дополнение, вычисление значения u_{n+1} требует только одного вычисления функции $f(x, u)$ не зависимо от порядка разностной схемы. Схемы Адамса могут быть построены следующим образом. Пусть $u(x)$ есть решение задачи (6.14). Введём обозначение $f(x, u(x)) = F(x)$, тогда

$$u(x_{n+1}) - u(x_n) = \int_{x_n}^{x_{n+1}} \frac{du}{dx} dx = \int_{x_n}^{x_{n+1}} F(x) dx.$$

Известно, что существует единственный полином $P_k(x)$ степени не выше чем k , принимающий в точках $x_n, x_{n-1}, \dots, x_{n-k}$ заданные значения $F(x_n), F(x_{n-1}), \dots, F(x_{n-k})$, соответственно (смотри Главу 7). Для достаточно гладкой функции $F(x)$, такой полином отклоняется от $F(x)$ на интервале $[x_n, x_{n+1}]$ на величину порядка h^{k+1} , то есть

$$\max_x |P_k(x) - F(x)| = O(h^{k+1}).$$

Тогда явные схемы Адамса имеют следующий вид

$$u_{n+1}^{(a)} - u_n^{(a)} = \int_{x_n}^{x_{n+1}} P_k(x) dx \quad (6.28)$$

Для $k=0$ полином

$$P_0(x) = f(x_n, u_n^{(a)}) = \text{const}$$

и выражение (6.28) превращается в (6.22) (схема Эйлера). Для $k=1$

$$P_1(x) = \frac{1}{h} [(x - x_{n-1})f(x_n, u_n^{(a)}) - (x - x_n)f(x_{n-1}, u_{n-1}^{(a)})],$$

и

$$\frac{1}{h} \int_{x_n}^{x_{n+1}} P_1(x) dx = f(x_n, u_n^{(a)}) + \frac{1}{2} [f(x_n, u_n^{(a)}) - f(x_{n-1}, u_{n-1}^{(a)})].$$

Аналогичным способом можно получить разностные схемы для $k = 2, 3, \dots$.

В общем, явные схемы Адамса могут быть представлены в следующем виде:

$$\frac{u_{n+1}^{(a)} - u_n^{(a)}}{h} = \sum_{k=0}^{p-1} a_k f(x_{n-k}, u_{n-k}^{(a)}), \quad (6.29)$$

$$n = p-1, \dots, N-1$$

где параметры a_k приведены в Таблице 6.3.

Таблица 6.3. Коэффициенты для явных схем Адамса.

Порядок аппроксимации, p	$a_k, k = 0, \dots, p-1$			
1	1			
2	3/2	-1/2		
3	23/12	-16/12	5/12	
4	55/24	-59/24	37/24	-9/24

Для того чтобы начать вычисления по схеме (6.29), когда $p \geq 2$, необходимо знать p значений

$$u_m^{(a)}, m = 0, \dots, p-1,$$

но только

$$u_0^{(a)} = \alpha$$

задано. Эти недостающие значения могут быть вычислены либо по схеме Рунге-Кутты соответствующего порядка, либо разложением решения в ряд Тейлора в точке $x=a$. Рассмотрим эту возможность более детально. Значение $u(x_m) = u(a+mh)$ вычисляется на основе следующего разложения

$$u(a + mh) = u(a) + mh \frac{du}{dx}(a) + \\ \frac{1}{2!} (mh)^2 \frac{d^2 u}{dx^2}(a) + \dots + \frac{1}{p!} (mh)^p \frac{d^p u}{dx^p}(a) + O(h^{p+1})$$

Используя уравнение (6.14), выразим производные функции $u(x)$ через функцию $f(x, u)$:

$$y_k(x, u) = \frac{d^k u}{dx^k} = \frac{dy_{k-1}}{dx} = \frac{\partial y_{k-1}}{\partial x} + \frac{\partial y_{k-1}}{\partial u} \frac{du}{dx} = \\ \frac{\partial y_{k-1}}{\partial x} + f \frac{\partial y_{k-1}}{\partial u} \\ y_1(x, u) = f(x, u), k = 2, \dots, p$$

Тогда начальные значения $u_m^{(a)}$ определяются по формуле

$$u_m^{(a)} = \alpha + \sum_{k=1}^p \frac{(mh)^k}{k!} y_k(a, \alpha), m = 1, \dots, p-1$$

Пример 6.5 (вычисление начальных значений в схеме Адамса)

Рассмотрим дифференциальную задачу

$$\frac{du}{dx} = -xu^2, x \geq 0 \\ u(0) = 1$$

Предположим мы хотим использовать схему Адамса четвертого порядка, поэтому нам нужно знать

$$u_1^{(a)}, u_2^{(a)} \text{ и } u_3^{(a)},$$

чтобы начать процесс вычислений. Производные $d^k u/dx^k$ выражаются следующим образом:

$$y_1(x, u) = -xu^2 \\ y_2(x, u) = -u^2 + 2x^2u^3 \\ y_3(x, u) = 6xu^3 - 6x^3u^4 \\ y_4(x, u) = 6u^3 - 36x^2u^4 + 24x^4u^5$$

Тогда начальные значения вычисляются по формулам

$$u_1^{(a)} = 1 - \frac{1}{2}h^2 + \frac{1}{4}h^4$$

$$u_2^{(a)} = 1 - 2h^2 + 4h^4$$

$$u_3^{(a)} = 1 - \frac{9}{2}h^2 + \frac{81}{4}h^4$$

Одной из особенностей методов Адамса является то, что для определения u_{n+1} , необходимо вычислить только значение $f(x_n, u_n)$, так как значения $f(x_{n-1}, u_{n-1})$, $f(x_{n-2}, u_{n-2})$, ... уже были вычислены в процессе определения $u_n, u_{n-1}, \dots$. Таким образом, преимущество схем Адамса над схемами Рунге-Кутты состоит в том, что они требуют меньшего количества операций на один шаг вычислительного процесса. Основные недостатки схем Адамса: требуется специальная процедура для вычисления дополнительных начальных значений, вычислительная схема усложняется при неравномерном шаге разностной сетки. Использование переменного шага необходимо в областях резкого изменения решения. Условия устойчивости явных схем Адамса, для модельного уравнения (6.16), приведены в Приложении С. Эти условия показывают что схемы Адамса имеют более строгие ограничения на шаг разностной сетки по сравнению со схемами Рунге-Кутты.

Если мы будем использовать точку x_{k+1} для построения полинома $P_k(x)$ в формуле (6.28), то получим неявные схемы Адамса. Они могут быть представлены в следующей форме:

$$\frac{u_{n+1}^{(a)} - u_n^{(a)}}{h} = \sum_{k=0}^{p-1} b_k f(x_{n+1-k}, u_{n+1-k}^{(a)})$$

$$n = \begin{cases} 0, \dots, N-1, & p = 1, 2 \\ p-2, \dots, N-1, & p = 3, 4, \dots \end{cases}, \quad (6.30)$$

где параметры b_k приведены в Таблице 6.4.

Неявные схемы Адамса дают более точные результаты, чем явные схемы того же порядка. В дополнение, неявные схемы имеют менее строгие ограничения на шаг разностной сетки по сравнению с явными схемами (схемы первого и второго порядка, и вовсе, безусловно устойчивы, смотри Приложение С).

Таблица 6.4. Коэффициенты для неявных схем Адамса.

Порядок аппроксимации, p	$b_k, k = 0, \dots, p-1$			
1	1			
2	1/2	1/2		
3	5/12	8/12	-1/12	
4	9/24	19/24	-5/24	1/24

Однако, неявные схемы не эффективны с точки зрения вычислительных затрат. В общем случае, невозможно выразить u_{n+1} из разностного уравнения (6.30). Вместо этого, мы должны решить нелинейное уравнение

$$u_{n+1}^{(a)} - b_0 h f(x_{n+1}, u_{n+1}^{(a)}) + \text{const} = 0$$

относительно u_{n+1} , используя тот или иной итерационный метод (смотри Главу 1), а это значительно увеличивает объём вычислений. Тем не менее, неявные схемы могут быть очень полезны в некоторых ситуациях, которые мы рассмотрим позднее.

Неявные схемы можно использовать для повышения точности приближённого решения полученного явным методом. Комбинация явной и неявной схемы называется методом предиктор-корректор. Общая процедура вычислений может быть организована следующим образом:

- 1) вначале, вычислим приближённое решение $\tilde{u}_{n+1}^{(a)}$, используя явную схему Адамса порядка p в качестве предиктора:

$$\tilde{u}_{n+1}^{(a)} = u_n^{(a)} + h \sum_{k=0}^{p-1} a_k f(x_{n-k}, u_{n-k}^{(a)})$$

- 2) затем, это приближение уточняется неявной схемой Адамса порядка p (корректор):

$$u_{n+1}^{(a)} = u_n^{(a)} + b_0 h f(x_{n+1}, \tilde{u}_{n+1}^{(a)}) + h \sum_{k=1}^{p-1} b_k f(x_{n+1-k}, u_{n+1-k}^{(a)})$$

Из построения видно, что схемы предиктор-корректор являются явными схемами и, следовательно, они условно устойчивы, но эти схемы имеют менее строгие ограничения на шаг разностной сетки по сравнению с явными схемами Адамса (смотри Приложение С).

6.3.4 Исследование устойчивости разностных схем в случае нелинейных задач

В параграфе 6.3.1 мы рассмотрели метод исследования устойчивости для модельного уравнения (6.16). Можно расширить возможности этого метода и применить его для анализа устойчивости разностных схем для уравнения (6.14). Предположим, что интегральная кривая уравнения (6.14) проходит через точку с координатами $x=x^*$, $u=u(x^*)$. Вблизи этой точки можно записать

$$f(x, u) \approx f(x^*, u^*) + \frac{\partial f}{\partial x}(x^*, u^*)(x - x^*) + \frac{\partial f}{\partial u}(x^*, u^*)(u - u^*)$$

и, следовательно, уравнение (6.14) можно приближённо заменить уравнением

$$\frac{du}{dx} = -\gamma^* u + \varphi(x),$$

где

$$\gamma^*(x^*, u^*) = \left| \frac{\partial f}{\partial u}(x^*, u^*) \right| = \text{const},$$

$$\varphi(x) = f(x^*, u^*) + \frac{\partial f}{\partial x}(x^*, u^*)(x - x^*) -$$

$$u^* \frac{\partial f}{\partial u}(x^*, u^*)$$

Здесь мы предполагаем, что дифференциальное уравнение (6.14) устойчиво, поэтому производная $\partial f / \partial u$ неположительная. Это уравнение выглядит как модельное уравнение (6.16) (функцией $\varphi(x)$ можно пренебречь, так как она не влияет на устойчивость), поэтому условия устойчивости разностных схем для модельного уравнения можно применить и для случая уравнения (6.14). Любая условио устойчивая схема для уравнения (6.16) имеет ограничение на выбор шага h : $h \leq \beta/\gamma$, $\beta = \text{const} > 0$ (смотри Приложение С). Естественно, что теперь значение коэффициента $\gamma = \gamma^*$ может меняться от точки к точке, поэтому необходимо модифицировать приведённое выше условие. Это значит, что

мы должны учитывать, что γ^* принимает не одно, а некоторое множество значений, которое описывает изменение $\partial f / \partial u$ вдоль интегральной кривой. По сути дела, существует два способа добиться устойчивости разностной схемы, а именно:

- 1) производить вычисления с переменным шагом h_n , который удовлетворяет условию

$$h_n \leq \beta / \gamma_n, \quad \gamma_n = \gamma^*(x_n, u_n^{(a)})$$

- 2) если мы можем оценить величину $u(x)$ на интервале интегрирования то шаг разностной сетки выбирается из условия

$$h \leq \beta / \max_{a \leq x \leq b} \gamma^*(x, u(x)),$$

и такое значение шага h обеспечивает устойчивость разностной схемы в любой точке интервала интегрирования.

В большинстве случаев встречающихся на практике, такой подход позволяет добиться устойчивых расчётов.

Пример 6.6 (анализ устойчивости разностной схемы для нелинейного уравнения)

Рассмотрим дифференциальную задачу

$$\begin{aligned} \frac{du}{dx} &= -\frac{10u^2}{x}, \quad 0.1 \leq x \leq 1.1 \\ u(0.1) &= 1 \end{aligned}$$

Решение этого уравнения ограничено: $u(x) \leq 1$. Тогда параметр γ^* можно оценить следующим образом:

$$\gamma^*(x, u) = \left| \frac{\partial f}{\partial u} \right| = \frac{20u}{x}.$$

Если мы будем использовать, например схему Рунге-Кутты второго порядка, то выбор

$$h \leq \frac{2}{\max_{0.1 \leq x \leq 1.1} \gamma^*(x, u)} = 0.01$$

обеспечивает устойчивость этой схемы в любой точке заданного интервала. Если мы нарушим это условие и выберем например $h=0.0159$, то полученное решение не является приближённым решением исходной задачи как показано на рисунке 6.3.

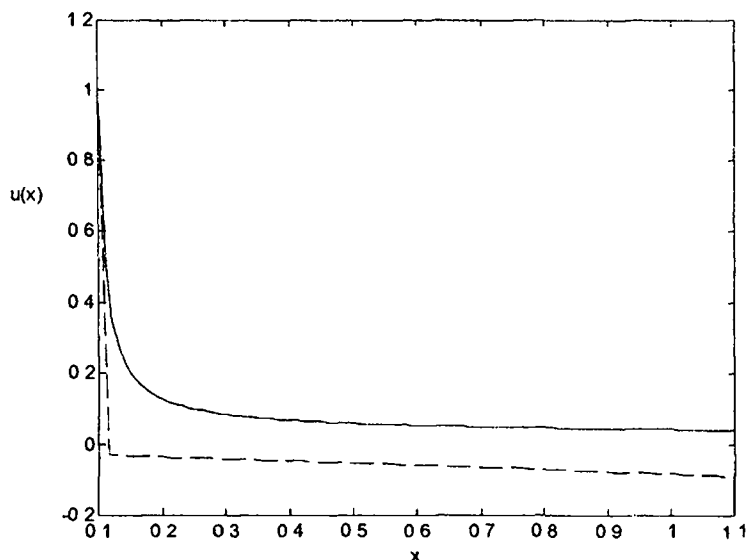


Рисунок 6.3. Пример неустойчивого решения,
 — - точное решение, — — — - приближенное решение

6.3.5 Системы дифференциальных уравнений

На практике мы часто сталкиваемся с задачами Коши не для одного уравнения, а для системы из N дифференциальных уравнений первого порядка. Все рассмотренные выше схемы для численного решения уравнения (6.14) непосредственно обобщаются на случай системы уравнений (6.15). Следует только заменить

$$u_n^{(a)} \text{ на } u_n^{(a)}$$

и

$$f(x, u_n^{(a)}) \text{ на } f(x, u_n^{(a)})$$

в схемах Рунге-Кутты и Адамса. Например, система уравнений

$$\begin{aligned}
 \frac{du_1}{dx} &= (1 - cu_2)u_1, \quad x \geq 0 \\
 \frac{du_2}{dx} &= (-1 + du_1)u_2 \\
 u_1(0) &= \alpha_1 \\
 u_2(0) &= \alpha_2
 \end{aligned} \tag{6.31}$$

может быть записана в форме

$$\begin{aligned}
 \frac{du}{dx} &= f(x, u), \quad x \geq 0 \\
 u(0) &= u_{in}
 \end{aligned}$$

если ввести обозначения

$$\begin{aligned}
 u &= \begin{pmatrix} u_1 \\ u_2 \end{pmatrix}, \\
 f(x, u) &= \begin{pmatrix} (1 - cu_2)u_1 \\ (-1 + du_1)u_2 \end{pmatrix}, \\
 u_{in} &= \begin{pmatrix} \alpha_1 \\ \alpha_2 \end{pmatrix}.
 \end{aligned}$$

Схема Рунге-Кутты (6.25) для системы уравнений имеет вид

$$\begin{aligned}
 u_{n+1}^{(a)} &= u_n^{(a)} + \frac{1}{2}h(k_1 + k_2) \\
 k_1 &= f(x_n, u_n^{(a)}) \\
 k_2 &= f(x_{n+1}, u_n^{(a)} + hk_1)
 \end{aligned}$$

и в случае системы (6.31) записывается как

$$\begin{pmatrix} u_{1,n+1}^{(a)} \\ u_{2,n+1}^{(a)} \end{pmatrix} = \begin{pmatrix} u_{1,n}^{(a)} \\ u_{2,n}^{(a)} \end{pmatrix} + \frac{1}{2}h \begin{pmatrix} k_{1,1} + k_{2,1} \\ k_{1,2} + k_{2,2} \end{pmatrix},$$

$$\begin{pmatrix} k_{11} \\ k_{12} \end{pmatrix} = \begin{pmatrix} (1 - cu_{2n}^{(a)})u_{1n}^{(a)} \\ (-1 + du_{1n}^{(a)})u_{2n}^{(a)} \end{pmatrix},$$

$$\begin{pmatrix} k_{21} \\ k_{22} \end{pmatrix} = \begin{pmatrix} (1 - c(u_{2n}^{(a)} + hk_{12}))(u_{1n}^{(a)} + hk_{11}) \\ -1 + d(u_{1n}^{(a)} + hk_{11}))(u_{2n}^{(a)} + hk_{12}) \end{pmatrix}.$$

Задача Коши для дифференциального уравнения вида

$$\frac{d^m u}{dx^m} = f(x, u, \frac{du}{dx}, \dots, \frac{d^{m-1}u}{dx^{m-1}}), a \leq x \leq b$$

$$\frac{d^k u}{dx^k}(a) = \alpha_k, k = 0, \dots, m-1$$

может быть также представлена в виде системы уравнений (6.15) путём замены переменных. Следующий пример демонстрирует эту процедуру

Уравнение

$$\frac{d^2 u}{dx^2} + \sin(x) \frac{du}{dx} + u^2 = 0, x \geq 0,$$

$$u(0) = \alpha_1,$$

$$\frac{du}{dx}(0) = \alpha_2,$$

можно представить в виде системы уравнений, если сделать замену

$$u_1(x) = u(x),$$

$$u_2(x) = \frac{du}{dx}.$$

В результате мы получим

$$\frac{du_1}{dx} = u_2,$$

$$\frac{du_2}{dx} = -\sin(xu_2 + u_1^2),$$

$$u_1(0) = \alpha_1,$$

$$u_2(0) = \alpha_2.$$

В параграфе 6.3.1 мы рассмотрели критерий устойчивости разностных схем для модельного уравнения (6.16). Теперь, в качестве модельной системы, выбирается следующая система уравнений:

$$\frac{du}{dx} = Au, x \geq 0,$$

где A - некоторая матрица с постоянными коэффициентами. Критерий устойчивости разностной схемы для этой модельной системы представляет собой некоторое ограничение на величину $w = h\lambda_m(A)$, где $\lambda_m(A)$ есть собственные значения матрицы A . Только, теперь появляется новая особенность. В общем случае, матрица A несимметричная, тогда её собственные значения не обязательно вещественные и, следовательно, параметр w может принимать комплексные значения. Поэтому разностная схема будет устойчивой, если параметр w (для всех значений m) принадлежит определённой области на комплексной плоскости. Такие области для схем Рунге-Кутты и Адамса показаны на рисунках 6.4-6.6.

В параграфе 6.3.4 мы рассмотрели, как можно обобщить критерий устойчивости, полученный для линейного модельного уравнения, на случай нелинейного уравнения. Этот критерий устанавливает некоторые ограничения на величину $h\gamma$, где γ оценивает скалярную величину $\partial f / \partial u$. В случае системы уравнений (6.15), естественным аналогом производной $\partial f / \partial u$ является матрица Якоби $F(x, u)$. Тогда параметр w определяемый теперь как $w = h\lambda_m(F)$ будет переменным. Разностная схема будет устойчивой, если шаг h выбирается таким образом, чтобы параметр w (для всех значений m) принадлежал определённой области на комплексной плоскости.

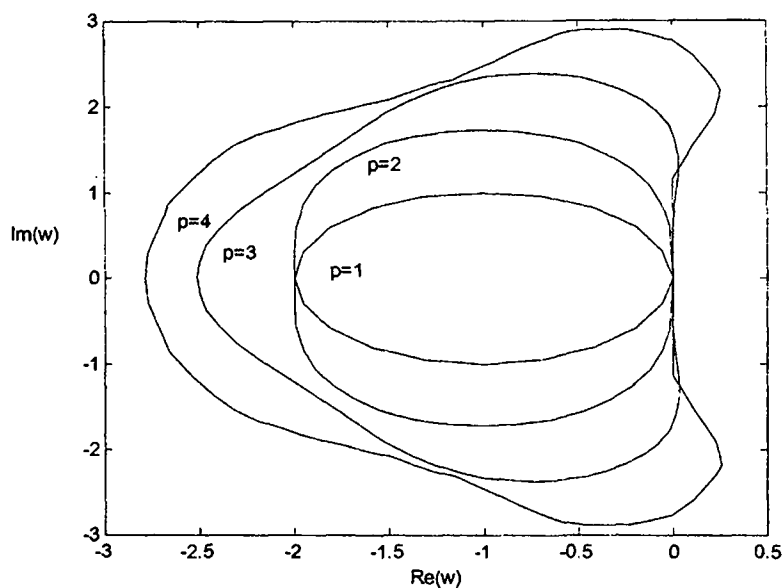


Рисунок 6.4. Области устойчивости на комплексной плоскости для p -стадийных явных схем Рунге-Кутты порядка $p=1, 2, 3, 4$. Область устойчивости лежит внутри контура.

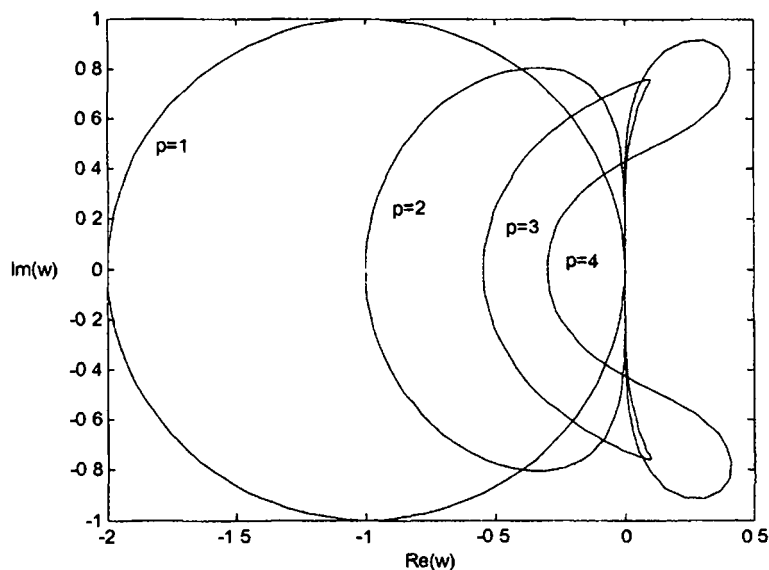


Рисунок 6.5. Области устойчивости на комплексной плоскости для явных схем Адамса порядка $p=1, 2, 3, 4$. Область устойчивости лежит внутри контура.

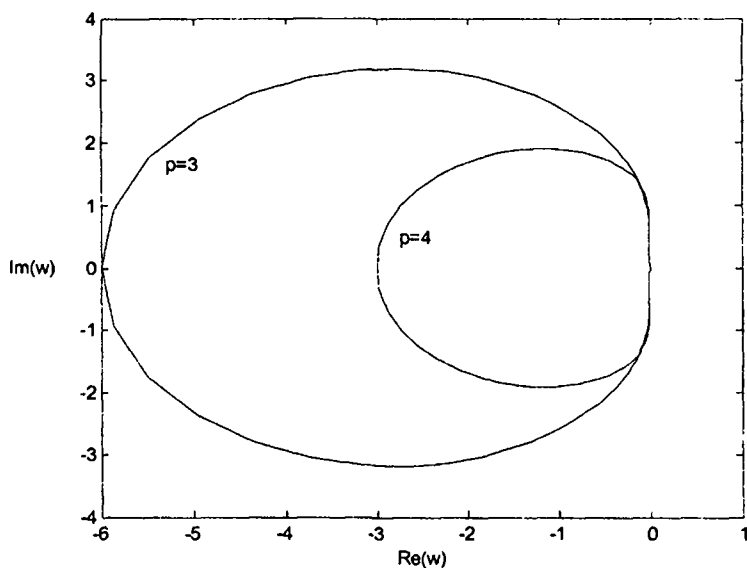


Рисунок 6.6. Области устойчивости на комплексной плоскости для неявных схем Адамса порядка $p=3, 4$ (схемы первого и второго порядка безусловно устойчивы). Область устойчивости лежит внутри контура.

6.3.6 Методы решения жёстких систем дифференциальных уравнений.

Во многих областях исследований возникают системы дифференциальных уравнений имеющие одно очень интересное свойство называемое «жёсткостью». Применение рассмотренных выше методов для численного решения таких систем, с вычислительной точки зрения, будет крайне неэффективным. Рассмотрим, например, следующую задачу Коши

$$\frac{du}{dx} = Au, \quad x \geq 0, \quad (6.32)$$

где

$$A = \begin{pmatrix} -21 & 19 & -20 \\ 19 & -21 & 20 \\ 40 & -40 & -40 \end{pmatrix}, \quad (6.33)$$

с начальными условиями

$$u(0) = \begin{pmatrix} 1 \\ 0 \\ -1 \end{pmatrix}.$$

Решение этой задачи имеет вид

$$\begin{aligned} u_1(x) &= \frac{1}{2} e^{-2x} + \frac{1}{2} e^{-40x} [\cos(40x) + \sin(40x)] \\ u_2(x) &= \frac{1}{2} e^{-2x} - \frac{1}{2} e^{-40x} [\cos(40x) + \sin(40x)] \\ u_3(x) &= -e^{-40x} [\cos(40x) - \sin(40x)] \end{aligned} \quad (6.34)$$

Графики $u_1(x)$, $u_2(x)$, и $u_3(x)$ представлены на рисунке 6.7.

На интервале $0 \leq x \leq 0.1$, все три компоненты изменяются очень быстро и при численном решении этой задачи, шаг разностной сетки должен быть очень мал. Для $x > 0.1$, однако, две компоненты решения, $u_1(x)$ и $u_2(x)$, практически совпадают и изменяются довольно медленно, а третья компонента, $u_3(x)$, близка к нулю. Поэтому на этом интервале

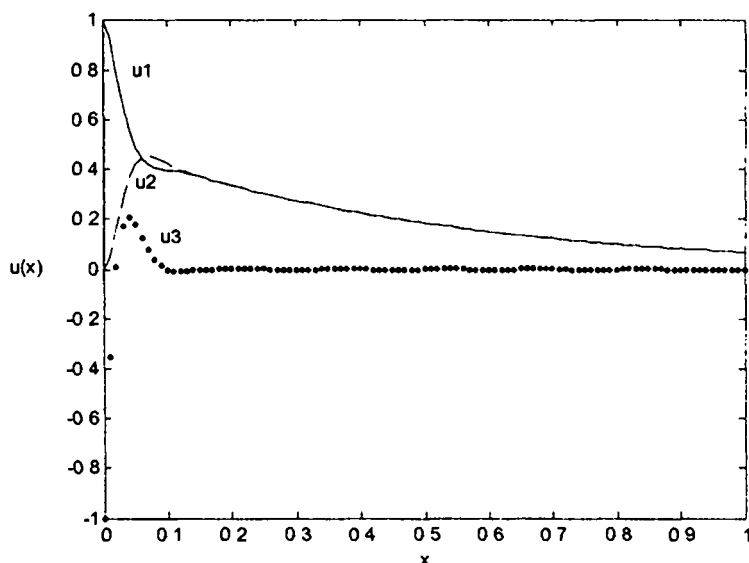


Рисунок 6.7. Точное решение задачи (6.32)

можно было бы задать и более крупный шаг разностной сетки. Матрицей Якоби $F(x, u)$ для уравнения (6.32) является матрица A определённая в (6.33) и её собственные значения λ_m равны $-2, -40 \pm 40i$. Если для решения системы (6.32) использовать какую-нибудь явную схему, то мы столкнёмся с довольно сильным ограничением на шаг h , так как последние два собственных значения имеют относительно большое по модулю значение. В тоже время, эти собственные значения определяют ту часть точного решения (6.34), которая пренебрежимо мала при $x \geq 0.1$. В результате, мы вынуждены задать очень маленький шаг разностной сетки на всём интервале интегрирования, что потребует большого объёма вычислений. Такая ситуация характерна для жёстких систем уравнений.

В случае нелинейной системы (6.15), жёсткость определяется собственными значениями матрицы Якоби $F(x, u)$, которые зависят от x . Система уравнений называется жёсткой на некотором интервале I , если

$$1) \operatorname{Re} \lambda_m(x) < 0, \quad m = 1, \dots, N,$$

$$2) \max_m |\operatorname{Re} \lambda_m(x)| \gg \min_m |\operatorname{Re} \lambda_m(x)|, \quad x \in I,$$

где $\lambda_m(x)$ - собственные значения матрицы $F(x, u)$. Отношение

$$\frac{\max_m |\operatorname{Re} \lambda_m(x)|}{\min_m |\operatorname{Re} \lambda_m(x)|}$$

называется коэффициентом жёсткости. Система уравнений (6.32) имеет коэффициент жёсткости равный 20, что вполне приемлемо, так как во многих практических задачах коэффициенты жёсткости зачастую достигают величины 10^6 .

Применение явных (условно устойчивых) схем для решения жёстких систем уравнений вызывает затруднение, так как это приводит к сильным ограничениям на величину шага разностной сетки. Единственный способ преодолеть это затруднение - применять неявные схемы, которые либо безусловно устойчивы когда $\operatorname{Re} \lambda_m < 0$, либо имеют большую область устойчивости. Со схемами первого рода мы уже встречались: это неявные схемы Адамса первого и второго порядка.

Другим популярным методом решения жёстких задач являются формулы дифференцирования назад (BDF, Backward Differentiation Formulae). Построение схем Адамса основано на интегрировании полинома, который интерполирует предыдущие значения $f(x, u)$. Построение схем BDF основано на дифференцировании полинома, который интерполирует предыдущие значения $u^{(a)}$. Полученная таким образом производная в точке x_n приравнивается затем значению $f(x_n, u_n)$. Схемы BDF порядка p для системы уравнений можно записать в следующем виде:

$$\frac{1}{h} \sum_{k=0}^p c_k u_{n+1-k}^{(a)} = d_0 f(x_{n+1}, u_{n+1}^{(a)}),$$

где соответствующие коэффициенты представлены в Таблице 6.5. Из этой формулы видно, что схемы BDF являются неявными схемами.

Области устойчивости для схем BDF показаны на рисунке 6.8. Схемы первого и второго порядка безусловно устойчивы когда $\operatorname{Re} \lambda_m < 0$, а схемы третьего и четвертого порядка имеют небольшие области неустойчивости для малых значений $\operatorname{Re} \lambda_m$, вблизи мнимой оси.

Таблица 6.5. Коэффициенты для схем BDF.

Порядок аппроксимации, p	d_0	$c_k, k = 0, \dots, p$			
1	1	1	-1		
2	2/3	1	-4/3	1/3	
3	6/11	1	-18/11	9/11	-2/11
4	12/25	1	-48/25	36/25	-16/25

Все разностные схемы пригодные для численного решения жёстких задач являются неявными схемами, поэтому на каждом шаге вычислений мы должны решить систему нелинейных уравнений вида

$$u_{n+1}^{(a)} = h \cdot \text{const} \cdot f(x_{n+1}, u_{n+1}^{(a)}) + r,$$

где r – некоторый известный вектор. Такая форма уравнений непосредственно подходит для применения метода простой итерации (4.3). Как было рассмотрено в параграфе 4.1, для сходимости метода простой итерации требуется выполнение условия

$$h \cdot \text{const} \cdot \|F(x, u)\| < 1. \quad (6.35)$$

Поэтому, имеет смысл применять метод простой итерации только для умеренно жёстких задач, когда $\|F(x, u)\|$ не очень велика. Для очень жёстких задач $\|F(x, u)\|$ велика и, согласно условию (6.35), мы сталкиваемся с сильным ограничением на величину шага разностной сетки. В этом случае, предпочтительнее использовать метод Ньютона и его модификации (смотри Главу 4), так как он обеспечивает сходимость итераций при более слабом ограничении на h чем условие (6.35).

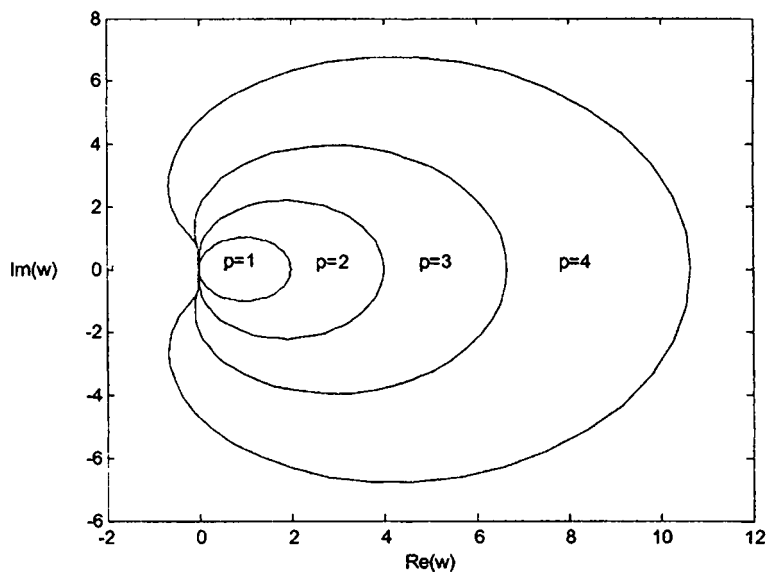


Рисунок 6.8. Области устойчивости на комплексной плоскости для схем BDF порядка $p=1, 2, 3, 4$. Область неустойчивости лежит внутри контура.

6.4. Численное решение краевых задач

В этом параграфе мы рассмотрим методы приближённого решения краевых задач, которые представляют собой дифференциальные уравнения с некоторыми условиями на решение, заданными в разных точках области определения этих задач. Мы познакомимся с основными методами численного решения краевых задач на примере дифференциального уравнения второго порядка вида

$$\frac{d^2 u}{dx^2} = f(x, u, \frac{du}{dx}), \quad a \leq x \leq b,$$

с условиями заданными в точках $x=a$ и $x=b$ (граничные условия). Мы будем предполагать, что это уравнение имеет следующие свойства:

- 1) функция $f(\dots)$ и её частные производные по u и du/dx непрерывны и,
- 2) частная производная функции $f(\dots)$ по u положительна, а частная производная по du/dx ограничена.

6.4.1. Метод стрельбы.

Рассмотрим следующую краевую задачу:

$$\begin{aligned} \frac{d^2 u}{dx^2} &= f(x, u, \frac{du}{dx}), \quad a \leq x \leq b, \\ u(a) &= \alpha \\ u(b) &= \beta \end{aligned} \tag{6.36}$$

В параграфе 6.3 мы рассмотрели различные методы решения задачи Коши. Можно применить эти методы для решения краевой задачи (6.36), но для этого нам надо изменить постановку задачи. Вместо краевой задачи (6.36) мы запишем следующую задачу Коши:

$$\frac{d^2 u}{dx^2} = f(x, u, \frac{du}{dx}), a \leq x \leq b,$$

$$u(a) = \alpha, \quad \frac{du}{dx}(a) = \operatorname{tg}(\varphi) \quad (6.37)$$

где φ обозначает угол между касательной к интегральной кривой в точке (a, α) и осью x (рисунок 6.9).

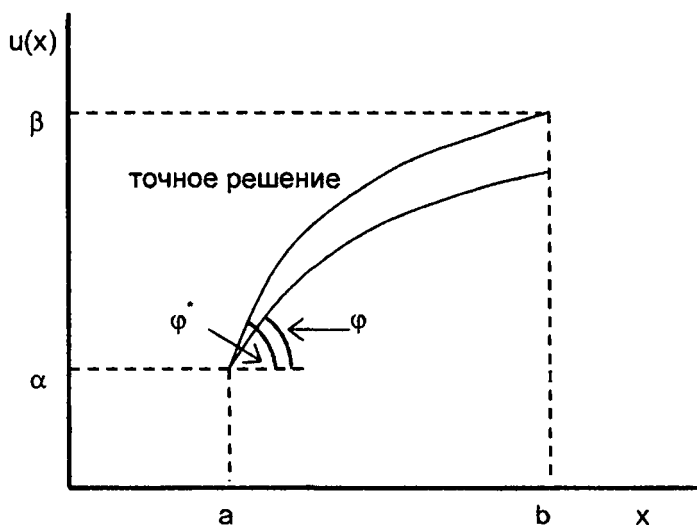


Рисунок 6.9. Графическое представление метода стрельбы.

Для заданного значения α , решение задачи (6.37) зависит от двух переменных, то есть $u=u(x, \varphi)$, и в точке $x=b$ это решение зависит только от φ . Тогда мы можем переформулировать задачу (6.37) следующим образом: найти такой угол $\varphi=\varphi^*$, чтобы кривая выходящая из точки (a, α) приходила в точку (b, β) , то есть,

$$u(b, \varphi^*) = \beta \quad (6.38)$$

При этом решение задачи (6.37) для такого угла $\varphi = \varphi^*$ будет совпадать с решением задачи (6.36) и всё сводится к решению уравнения (6.38). Это уравнение имеет стандартный вид $f(\varphi) = 0$, где $f(\varphi) = u(b, \varphi) - \beta$. Правда, оно отличается от стандартных уравнений тем, что функция $f(\varphi)$ не описывается аналитическим выражением, а определяется через приближённое решение задачи (6.37). Замена краевой задачи на сформулированную специальным образом задачу Коши является основной особенностью метода стрельбы.

Для решения уравнения (6.38) можно использовать различные методы (смотри Главу 1). Например, если мы найдём такие значения углов φ_0 и φ_1 , что разности $u(b, \varphi_0) - \beta$ и $u(b, \varphi_1) - \beta$ имеют противоположные знаки, то далее мы можем применить метод деления отрезка пополам. При использовании метода секущих, мы сначала должны задать некоторые значения φ_0 и φ_1 , и последующие значения φ_k затем вычисляются по формуле

$$\varphi_{k+1} = \varphi_k - \frac{f(\varphi_k)(\varphi_k - \varphi_{k-1})}{f(\varphi_k) - f(\varphi_{k-1})}, \quad k = 1, 2, \dots,$$

Эти вычисления продолжаются до тех пор, пока не выполнится условие

$$|u(b, \varphi_k) - \beta| \leq \varepsilon_p,$$

где ε_p - требуемая точность приближённого решения.

Метод стрельбы хорошо работает в тех случаях, когда решение $u(x, \varphi)$ не очень «сильно» зависит от φ . В противном случае он становится неустойчивым. Следующий пример поясняет что значит «сильная» зависимость от φ .

Пример 6.7 (неустойчивость метода стрельбы)

Рассмотрим следующую краевую задачу:

$$\begin{aligned} \frac{d^2 u}{dx^2} &= \gamma^2 u, \quad 0 \leq x \leq 1, \\ u(0) &= \alpha, \\ u(1) &= \beta, \end{aligned} \quad (6.39)$$

где γ - некоторая константа. Решение этой задачи имеет вид

$$u(x) = \frac{\exp(-\gamma x) - \exp(-\gamma(2-x))}{1 - \exp(-2\gamma)} \alpha + \frac{\exp(-\gamma(1-x)) - \exp(-\gamma(1+x))}{1 - \exp(-2\gamma)} \beta$$

Коэффициенты при α и β ограничены на отрезке $0 \leq x \leq 1$: для всех значений $\gamma > 0$ они не превышают единицу. Следовательно, небольшая ошибка в задании α и β приводит к небольшой ошибке в решении. Рассмотрим теперь задачу Коши

$$\begin{aligned} \frac{d^2 u}{dx^2} &= \gamma^2 u, \quad 0 \leq x \leq 1, \\ u(0) &= \alpha, \\ \frac{du}{dx}(0) &= \tan(\varphi). \end{aligned}$$

Решение этой задачи имеет вид

$$u(x) = \frac{1}{2\gamma} (\alpha\gamma + \tan(\varphi)) \exp(\gamma x) + \frac{1}{2\gamma} (\alpha\gamma - \tan(\varphi)) \exp(-\gamma x)$$

Предположим мы задали величину $\tan(\varphi)$ с точностью ε , тогда решение в точке $x=1$ будет отличаться от точного значения на величину

$$\Delta u(1) = \frac{\varepsilon}{2\gamma} \exp(\gamma) - \frac{\varepsilon}{2\gamma} \exp(-\gamma).$$

Хорошо видно, что для больших значений γ это отличие становится недопустимо большим, даже если ε достаточно мало. Поэтому, не имеет смысла применять метод стрельбы для решения задачи (6.39) при больших значениях γ .

6.4.2 Сведение разностной схемы к системе уравнений.

В предыдущем параграфе мы обсудили применение численных методов для задачи Коши к решению некоторых краевых задач. Далее мы рассмотрим построение разностных схем с использованием разностных отношений, полученных в параграфе 6.2.2. Начнём с линейной краевой задачи второго порядка, имеющей следующий вид

$$\begin{aligned} \frac{d^2 u}{dx^2} &= p(x) \frac{du}{dx} + q(x)u + r(x), \quad a \leq x \leq b \\ u(a) &= \alpha \\ u(b) &= \beta \end{aligned} \quad (6.40)$$

Введём разностную сетку с узлами $x_n = a + nh$, $n=0, \dots, N$, где $h=(b-a)/N$ - шаг сетки. Теперь мы заменим производные в (6.40) приближёнными разностными отношениями

$$\begin{aligned} \frac{d^2 u}{dx^2}(x_n) &\approx \frac{u(x_n + h) - 2u(x_n) + u(x_n - h))}{h^2} \\ \frac{du}{dx}(x_n) &\approx \frac{u(x_n + h) - u(x_n - h))}{2h} \end{aligned} \quad (6.41)$$

В результате мы получим следующую разностную схему:

$$\begin{aligned} \frac{1}{h^2} (u_{n+1}^{(a)} - 2u_n^{(a)} + u_{n-1}^{(a)}) &= \\ \frac{p_n}{2h} (u_{n+1}^{(a)} - u_{n-1}^{(a)}) + q_n u_n^{(a)} + r_n, \quad n = 1, \dots, N-1 \\ u_0^{(a)} &= \alpha \\ u_N^{(a)} &= \beta \end{aligned}$$

Эта схема аппроксимирует задачу (6.40) с порядком h^2 , так как разностные выражения (6.41) аппроксимируют производные с порядком h^2 , а граничные условия выполняются точно. После приведения подобных слагаемых, схему можно записать в виде

$$\begin{aligned}
 u_0^{(a)} &= \alpha \\
 (1 + \frac{hp_n}{2})u_{n-1}^{(a)} - (2 + h^2q_n)u_n^{(a)} + (1 - \frac{hp_n}{2})u_{n+1}^{(a)} &= h^2r_n, \\
 n &= 1, \dots, N-1 \\
 u_N^{(a)} &= \beta
 \end{aligned} \tag{6.42}$$

Легко заметить, что эти выражения определяют систему линейных уравнений с трёхдиагональной матрицей (смотри (2.9)). Если $h|p_n|/2 < 1$ и $q_n > 0$, тогда условия (2.11) выполняются и можно использовать метод прогонки для решения системы (6.42). Решая эту систему, мы получим $\{u^{(a)}\}$, приближённое решение во всех узлах разностной сетки.

Построение разностной схемы для нелинейной краевой задачи вида

$$\begin{aligned}
 \frac{d^2 u}{dx^2} &= f(x, u, \frac{du}{dx}), a \leq x \leq b, \\
 u(a) &= \alpha \\
 u(b) &= \beta
 \end{aligned} \tag{6.43}$$

производится аналогичным образом. Однако, система уравнений в этом случае будет нелинейной, что потребует применения какого-нибудь итерационного метода для её решения. Если заменить производные в (6.43) приближёнными разностными отношениями (6.41), то разностная схема будет иметь вид

$$\begin{aligned}
 \frac{1}{h^2} (u_{n+1}^{(a)} - 2u_n^{(a)} + u_{n-1}^{(a)}) &= \\
 f\left(x_n, u_n^{(a)}, \frac{u_{n+1}^{(a)} - u_{n-1}^{(a)}}{2h}\right), n &= 1, \dots, N-1 \\
 u_0^{(a)} &= \alpha \\
 u_N^{(a)} &= \beta
 \end{aligned} \tag{6.44}$$

Эту схему можно представить в виде системы нелинейных уравнений

$$v_0 = \alpha$$

$$v_0 - 2v_1 + v_2 - h^2 f\left(x_1, v_0, \frac{v_2 - v_0}{2h}\right) = 0 \quad (6.45)$$

.....

$$v_{N-2} - 2v_{N-1} + v_N - h^2 f\left(x_{N-1}, v_{N-1}, \frac{v_N - v_{N-2}}{2h}\right) = 0$$

$$v_N = \beta$$

Решение этой системы $\mathbf{v}=(v_0, \dots, v_N)$ даёт приближённое значение $u^{(a)}$. Для решения системы (6.45) можно использовать различные итерационные методы, рассмотренные в Главе 4. Если применить, например, метод Ньютона, то на каждой итерации необходимо решать систему линейных уравнений с матрицей Якоби. В данном случае, матрица Якоби является трехдиагональной, поэтому можно применить либо метод прогонки, либо соответствующую процедуру из библиотеки программ. Начальное приближение $\mathbf{v}^{(0)}$ можно выбрать в виде линейной функции проходящей через точки (a, α) и (b, β) :

$$v_n^{(0)} = \frac{\beta - \alpha}{b - a} x_n + \frac{\alpha b - \beta a}{b - a}, n = 0, \dots, N$$

Пример 6.8 (нелинейная краевая задача)

Рассмотрим следующую краевую задачу:

$$\begin{aligned} \frac{d^2 u}{dx^2} &= -\exp(-u) \frac{du}{dx}, 0 \leq x \leq 1 \\ u(0) &= 0 \\ u(1) &= \ln(2) \end{aligned} \quad (6.46)$$

Разностная схема (6.44) для этой задачи записывается как

$$\begin{aligned} \frac{1}{h^2} (u_{n+1}^{(a)} - 2u_n^{(a)} + u_{n-1}^{(a)}) + \frac{\exp(-u_n^{(a)})}{2h} (u_{n+1}^{(a)} - u_{n-1}^{(a)}) &= 0, \\ n &= 1, \dots, N-1 \\ u_0^{(a)} &= 0 \\ u_N^{(a)} &= \ln(2) \end{aligned}$$

Тогда соответствующая система нелинейных уравнений имеет вид

$$v_0 = \alpha$$

$$v_{n-1} \left(1 - \frac{1}{2} h \cdot \exp(-v_n) \right) - 2v_n +$$

$$v_{n+1} \left(1 + \frac{1}{2} h \cdot \exp(-v_n) \right) = 0, n = 1, \dots, N-1$$

$$v_N = \beta$$

Если выбрать начальное приближение

$$v_n^{(0)} = x_n \ln(2), n = 0, \dots, N$$

и относительную точность 10^{-5} , то потребуется всего три итерации по методу Ньютона, для того чтобы решить эту систему. Относительная ошибка полученного приближённого решения для $h=0.1$ равна

$$\frac{\|u^{(e)} - v\|_{\infty}}{\|u^{(e)}\|_{\infty}} \approx 7.8 \cdot 10^{-5}$$

6.4.3 Метод последовательных приближений.

Идея метода последовательных приближений состоит в том, что решение нелинейной задачи сводится к решению набора линейных задач. Предположим, мы знаем некоторую функцию $v_0(x)$, которая удовлетворяет граничным условиям и является грубым приближением к точному решению $u(x)$. Пусть

$$u(x) \approx v_0(x) + w_0(x), \quad (6.47)$$

где $w_0(x)$ малая поправка к функции $v_0(x)$. Подставим выражение (6.47) в уравнение (6.43) и проведём линеаризацию:

$$\frac{d^2 u}{dx^2} = \frac{d^2 v_0}{dx^2} + \frac{d^2 w_0}{dx^2},$$

$$\begin{aligned}
f\left(x, v_0 + w_0, \frac{dv_0}{dx} + \frac{dw_0}{dx}\right) &= f\left(x, v_0, \frac{dv_0}{dx}\right) + \\
\frac{\partial f}{\partial u}\left(x, v_0, \frac{dv_0}{dx}\right)w_0 &+ \frac{\partial f}{\partial\left(\frac{du}{dx}\right)}\left(x, v_0, \frac{dv_0}{dx}\right)\frac{dw_0}{dx} + \\
O(w_0^2 + (\frac{dw_0}{dx})^2)
\end{aligned}$$

Пренебрегая слагаемым $O(\dots)$, мы получим линейную задачу для поправки $w_0(x)$:

$$\begin{aligned}
\frac{d^2 w_0}{dx^2} &= p(x)\frac{dw_0}{dx} + q(x)w_0 + r(x), \\
a \leq x \leq b \\
w_0(a) = w_0(b) &= 0
\end{aligned} \tag{6.48}$$

где

$$\begin{aligned}
p(x) &= \frac{\partial f}{\partial\left(\frac{du}{dx}\right)}\left(x, v_0, \frac{dv_0}{dx}\right), \\
q(x) &= \frac{\partial f}{\partial u}\left(x, v_0, \frac{dv_0}{dx}\right), \\
r(x) &= f\left(x, v_0, \frac{dv_0}{dx}\right) - \frac{d^2 v_0}{dx^2}.
\end{aligned}$$

Решая линейную задачу (6.48) мы найдём поправку $w_0(x)$, тогда $v_1(x) = v_0(x) + w_0(x)$ будет следующим приближением к решению $u(x)$. Таким образом, рассмотренная процедура определяет последовательность линейных краевых задач, решения которых сходятся к решению исходной нелинейной краевой задачи. Хотя в общем случае доказать наличие сходимости довольно затруднительно тем не менее этот метод может использоваться для получения хороших начальных приближений при решении нелинейных систем типа (6.45).

Пример 6.49 (метод последовательных приближений)

Рассмотрим задачу (6.46). Если мы возьмём начальное решение $v_0(x) = x \ln(2)$, тогда уравнение (6.48) примет вид ($\gamma = \ln(2)$)

$$\frac{d^2 w_0}{dx^2} = -\exp(-\gamma x) \frac{dw_0}{dx} + \gamma \exp(-\gamma x) w_0 - \gamma \exp(-\gamma x),$$

$$0 \leq x \leq 1$$

$$w_0(0) = w_0(1) = 0$$

Численно решая это уравнение для $h=0.1$, мы получим сеточную функцию

$$w_0^{(a)} = \{w_{0,n}\},$$

показанную на рисунке 6.10.

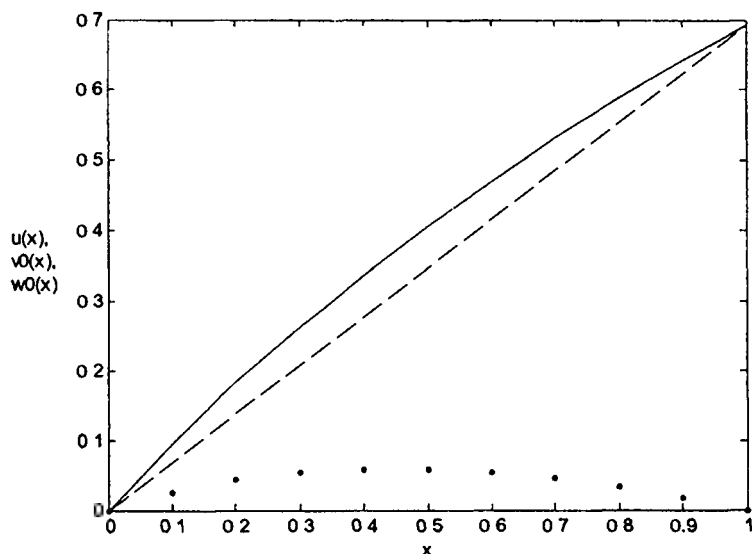


Рисунок 6.10. Приближённое решение задачи (6.46); — - точное решение; — - $v_0(x)$; • - $w_0(x)$.

Следующее приближение определяется как

$$\mathbf{v}_1^{(a)} = \mathbf{v}_0^{(a)} + \mathbf{w}_0^{(a)} = \{x_n \ln(2) + w_{0,n}^{(a)}\}$$

и ошибка этого приближения равна

$$\frac{\|\mathbf{u}^{(e)} - \mathbf{v}_1^{(a)}\|_{\infty}}{\|\mathbf{u}^{(e)}\|_{\infty}} \approx 2.3 \cdot 10^{-4}.$$

Эти результаты показывают, что в данном случае, решение линеаризованной задачи даёт довольно хорошее приближение к решению исходной нелинейной задачи.

6.4.4. Метод установления

Решение краевой задачи можно трактовать как равновесное состояние, к которому приближается решение некоторой нестационарной задачи. Иногда возникает ситуация когда удобнее и эффективнее, с вычислительной точки зрения, решать такую нестационарную задачу, чем непосредственно искать решение исходной краевой задачи. Такой подход называется методом установления. Мы обсудим основные особенности этого метода в процессе построения численного алгоритма для решения следующей краевой задачи:

$$\begin{aligned} \frac{d^2 u}{dx^2} &= c_1(x, u) \frac{du}{dx} + c_2(x)u, a \leq x \leq b \\ g_1\left(u(a), \frac{du}{dx}(a)\right) &= 0 \\ g_2\left(u(b), \frac{du}{dx}(b)\right) &= 0 \end{aligned} \tag{6.49}$$

где $c_1(x, u)$, $c_2(x)$, $g_1(\dots)$, $g_2(\dots)$ некоторые заданные функции. Вначале мы должны сформулировать некоторую нестационарную задачу. В нашем случае она может быть записана в следующей форме:

$$\begin{aligned}
\frac{\partial v}{\partial t} &= \frac{\partial^2 v}{\partial x^2} - c_1(x, v) \frac{\partial v}{\partial x} - c_2(x) v, \\
a &\leq x \leq b, t \geq 0 \\
g_1 \left(v(a), \frac{\partial v}{\partial x}(a) \right) &= 0 \\
g_2 \left(v(b), \frac{\partial v}{\partial x}(b) \right) &= 0 \\
v(0, x) &= g_0(x)
\end{aligned} \tag{6.50}$$

где функция $g_0(x)$ задаёт произвольные начальные условия. Так как граничные условия не зависят от времени, то естественно ожидать, что решение $v(x, t)$ с течением времени будет меняться всё медленнее и медленнее. Поэтому в пределе $t \rightarrow \infty$, это решение будет стремиться к решению задачи (6.49), то есть

$$\lim_{t \rightarrow \infty} v(x, t) = u(x).$$

Следовательно, вместо стационарной задачи (6.49) можно решать нестационарную задачу (6.50) до некоторого момента времени $\bar{t}$, когда решение $v(x, t)$ перестанет изменяться в пределах заданной точности. Это и есть основная идея метода установления.

В соответствии с нашим рассмотрением нам теперь нужно построить разностную схему для задачи (6.50). Вначале мы должны задать разностную сетку. Для одномерных нестационарных уравнений она определяется узлами (x_n, t_k) как показано на рисунке 6.11. Параметры $h = x_{n+1} - x_n$ и $\tau_k = t_{k+1} - t_k$ называются соответственно шагами по пространству и времени.

Как и прежде заменим производные по пространству и времени в уравнении (6.50) разностными отношениями. В результате мы получим следующую схему:

$$\begin{aligned}
\frac{v_n^{k+1} - v_n^k}{\tau_k} &= \frac{v_{n+1}^k - 2v_n^k + v_{n-1}^k}{h^2} - c_1(x_n, v_n^k) \frac{v_{n+1}^k - v_{n-1}^k}{2h} - \\
&\quad c_2(x_n) v_n^k, \quad n = 1, \dots, N-1; k = 0, 1, \dots
\end{aligned} \tag{6.51}$$

$$g_1^*(v_0^{k+1}, v_1^{k+1}) = 0$$

$$g_2^*(v_{N-1}^{k+1}, v_N^{k+1}) = 0,$$

$$v_n^0 = g_0(x_n)$$

здесь верхний индекс k обозначает момент времени t_k .

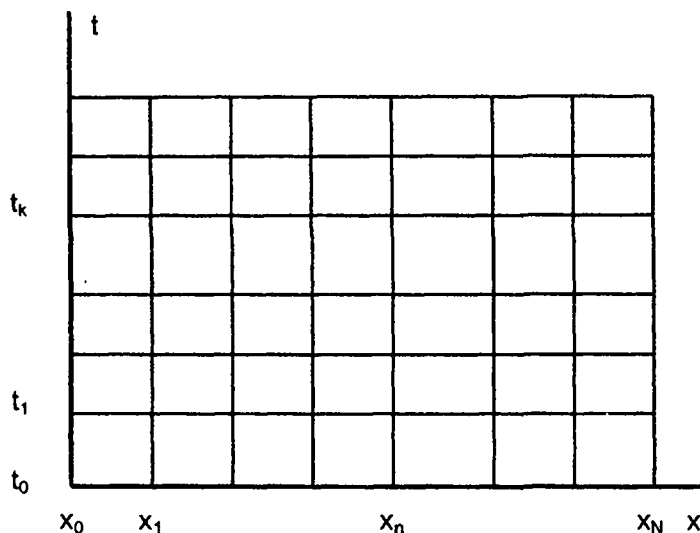


Рисунок 6.11. Разностная сетка на плоскости (x, t) .

Для вычисления приближённого решения, перепишем разностную схему (6.51) в виде

$$v_n^{k+1} = \left(1 - \frac{2\tau_k}{h^2} - \tau_k c_2(x_n)\right) v_n^k + \left(\frac{\tau_k}{h^2} - \frac{\tau_k c_1(x_n, v_n^k)}{2h}\right) v_{n+1}^k + \left(\frac{\tau_k}{h^2} + \frac{\tau_k c_1(x_n, v_n^k)}{2h}\right) v_{n-1}^k, \quad (6.52)$$

$$n = 1, \dots, N-1; k = 0, 1, \dots$$

$$g_1^*(v_0^{k+1}, v_1^{k+1}) = 0$$

$$g_2^*(v_{N-1}^{k+1}, v_N^{k+1}) = 0$$

$$v_n^0 = g_0(x_n)$$

Из начального условия $v(x, 0) = g_0(x)$ следует, что значения $v^0 = \{g_0(x_n)\}$ известны. Поэтому, из формулы (6.52) можно вычислить значения $v^1 = \{v(x_n, t_1)\}$. Повторяя эту процедуру, после того как значения v^1 определены, можно последовательно вычислить значения $v^2, v^3, \dots$

Очевидно, что построенная нами схема является явной схемой и значит она условно устойчивая. Методы анализа устойчивости такого рода схем требуют специального рассмотрения, что выходит за рамки этой книги, поэтому далее приводится окончательный результат. Разностная схема (6.51) будет устойчива если шаг по времени выбирается исходя из условия

$$\tau_k \leq \min(\tau^{(1)}, \tau_k^{(2)}),$$

где

$$\tau^{(1)} = \frac{2h^2}{4 + h^2 \max_x c_2(x)},$$

$$\tau_k^{(2)} = \frac{2h^2(2 + h^2 \max_x c_2(x))}{h^2 \max_n (c_1(x_n, v_n^k))^2 + (2 + h^2 \max_x c_2(x))^2}$$

Метод установления является простым и надёжным методом, так как он даёт приближённое решение, которое сходится к точному решению стационарной задачи при любом начальном условии. Когда приближённое решение v^k уже не изменяется в пределах заданной точности, то есть

$$\frac{\|v^{k+1} - v^k\|_\infty}{\|v^{k+1}\|_\infty} \leq \varepsilon_p,$$

процесс установления завершён.

Пример 6.10 (метод установления)

Как и прежде, мы рассмотрим задачу (6.46). Тогда соответствующая нестационарная задача имеет вид

$$\frac{\partial v}{\partial t} = \frac{\partial^2 v}{\partial x^2} - \exp(-v) \frac{\partial v}{\partial x}, 0 \leq x \leq 1, t \geq 0$$

$$v(0) = 0$$

$$v(1) = \ln(2)$$

Для примера, выберем начальную функцию $g_0(x)=0$, которая не удовлетворяет второму граничному условию и довольно далека от точного решения. Процесс установления показан на рисунке 6.12. Если задать $\varepsilon_p=10^{-4}$, то потребуется 134 шагов по времени для завершения этого процесса и относительная ошибка полученного решения равна

$$\frac{\|u^{(e)} - v^{134}\|_{\infty}}{\|u^{(e)}\|_{\infty}} \approx 8 \cdot 10^{-4}.$$

Если выбрать лучшее начальное условие (например, $v_0(x)=x\ln(2)$), то потребуется гораздо меньше шагов по времени чтобы получить требуемое приближённое решение.

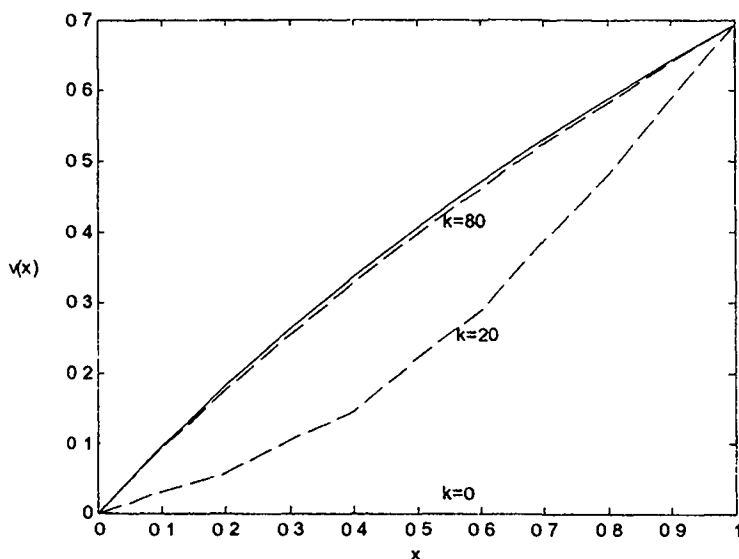


Рисунок 6.12. Последовательные приближения, полученные методом установления (задача (6.46), $h = 0.1, \tau = h^2 / 2$); — — — точное решение; — — — приближённые решения.

6.4.5 Аппроксимация граничных условий в случае, когда на границе задано значение производной.

До сих пор мы рассматривали краевые задачи с граничными условиями первого рода, когда решение принимает некоторые заданные значения в граничных точках. Моделирование различных физических процессов часто приводит к краевым задачам, где в граничных точках задаются значения производных решения (например, граничные условия второго и третьего рода для дифференциальных уравнений второго порядка). В качестве примера рассмотрим построение разностной схемы для следующей краевой задачи:

$$\begin{aligned}\frac{d^2 u}{dx^2} &= f\left(x, u, \frac{du}{dx}\right), a \leq x \leq b \\ \frac{du}{dx}(a) &= \alpha \\ u(b) &= \beta\end{aligned}\tag{6.53}$$

Разностная аппроксимация дифференциального оператора аналогична выражению (6.44):

$$\begin{aligned}\frac{1}{h^2}(u_{n+1}^{(a)} - 2u_n^{(a)} + u_{n-1}^{(a)}) &= f\left(x_n, u_n^{(a)}, \frac{u_{n+1}^{(a)} - u_{n-1}^{(a)}}{2h}\right), \\ h &= 1, \dots, N-1 \\ u_N^{(a)} &= \beta\end{aligned}\tag{6.54}$$

Очевидный способ аппроксимировать du/dx в точке a - использовать разность вперёд

$$\frac{u_1^{(a)} - u_0^{(a)}}{h} = \alpha\tag{6.55}$$

Однако это выражение аппроксимирует производную в точке a с порядком h , в то время как выражение (6.54) аппроксимирует дифференциальное уравнение с порядком h^2 . Это значит что разностная схема (6.54) и (6.55) имеет аппроксимацию только первого порядка. Таким образом, менее точная аппроксимация лишь в одной точке снижает точность всего решения. Поэтому необходимо построить более

точное приближение для производной в граничной точке. Для этого введём фиктивный узел $x_{-1}=a-h$, что даёт нам возможность использовать центральную разность для аппроксимации граничного условия $du/dx=\alpha$ с порядком h^2 :

$$\frac{u_1^{(a)} - u_{-1}^{(a)}}{2h} = \alpha \quad (6.56)$$

Для того чтобы исключить неизвестную величину $u_{-1}^{(a)}$, запишем выражение (6.54) для $n=0$:

$$\frac{1}{h^2} (u_1^{(a)} - 2u_0^{(a)} + u_{-1}^{(a)}) = f\left(x_0, u_0^{(a)}, \frac{u_1^{(a)} - u_{-1}^{(a)}}{2h}\right)$$

После подстановки (6.56) в предыдущее выражение мы окончательно получим

$$u_1^{(a)} - u_0^{(a)} - \frac{1}{2}h^2 f(a, u_0^{(a)}, \alpha) = \alpha h \quad (6.57)$$

Для вычисления приближённого решения задачи (6.53), необходимо решить систему (6.45), предварительно заменив первое уравнение в этой системе на уравнение

$$v_1 - v_0 - \frac{1}{2}h^2 f(a, v_0, \alpha) = \alpha h.$$

Рассмотренный нами пример является простейшей реализацией более общего подхода, который называется методом фиктивных областей. Этот метод успешно используется в различных задачах для аппроксимации производных в граничных точках.

Пример 6.11 (аппроксимация граничных условий второго рода)

Рассмотрим следующую простую задачу:

$$\begin{aligned} \frac{d^2 u}{dx^2} &= u, 0 \leq x \leq 1 \\ \frac{du}{dx}(0) &= -1 \\ u(1) &= 1 \end{aligned} \quad (6.58)$$

Если мы используем выражение (6.55) для аппроксимации граничного условия в точке $x=0$, то приближённое решение будет заметно отличаться от точного решения, как показано на рисунке 6.13, с относительной ошибкой

$$e_r = \frac{\|u^{(e)} - u^{(a)}\|_{\infty}}{\|u^{(e)}\|_{\infty}} = 2.7 \cdot 10^{-2}$$

Более точная аппроксимация (6.57) позволяет значительно уменьшить эту ошибку: в этом случае $e_r = 4.1 \cdot 10^{-4}$.

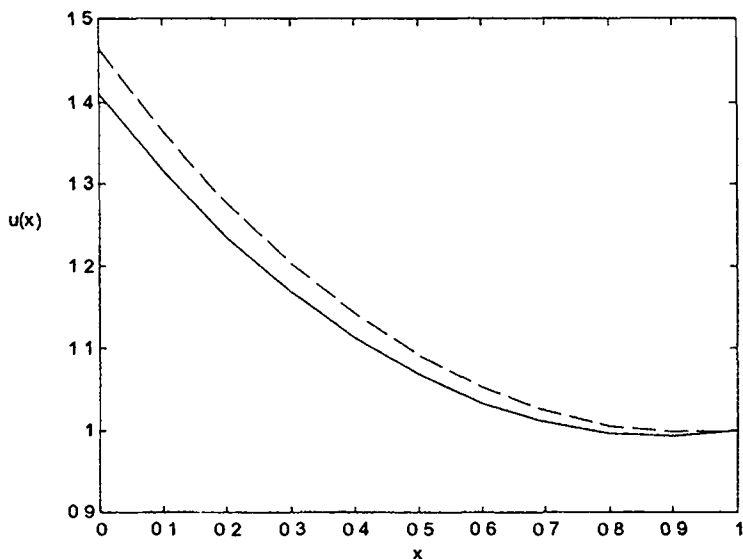


Рисунок 6.13. Приближённое решение задачи (6.58); — — — - точное решение; — — — - приближённое решение. Для аппроксимации граничного условия использовалось выражение (6.55).

6.5 Оценка ошибки приближённого решения.

В этом параграфе мы рассмотрим несколько методов оценки и контроля ошибки приближённого решения. Имея возможность оценить локальную или глобальную ошибку приближённого решения, можно подобрать шаг сетки таким образом, чтобы эта ошибка была меньше заданной величины, которую мы будем обозначать ϵ_r . Основная идея рассматриваемых ниже методов состоит в том, что для оценки ошибки используются два различных приближённых решения.

Мы начнём наше обсуждение с оценки ошибки в случае численного решения задачи Коши. Вычислим два решения

$$U_n^{(a)} \text{ и } \tilde{U}_n^{(a)}$$

в точке x_n , где $\tilde{U}_n^{(a)}$ обозначает более точное решение. Тогда разница

$$U_n^{(a)} - \tilde{U}_n^{(a)}$$

даёт оценку локальной ошибки

$$e_n = U_n^{(e)} - U_n^{(a)}$$

для менее точного решения. Пара схем Рунге-Кутты с порядками аппроксимации p и $p+1$ (так называемая вложенная пара) может быть использована для вычисления двух приближённых решений. Вложенные пары строятся таким образом, чтобы вычисление функции производилось в одних и тех же точках для обеих схем, что значительно уменьшает число вычислений. Так s -этапная вложенная пара может быть представлена в следующем виде

0	0				
c_2	$a_{2,1}$				
c_3	$a_{3,1}$	$a_{3,2}$			
.	.	.	.		
c_s	$a_{s,1}$	$a_{s,2}$	...	$a_{s,s-1}$	
	$b_1^{(1)}$	$b_2^{(1)}$	...	$b_{s-1}^{(1)}$	$b_s^{(1)}$
	$b_1^{(2)}$	$b_2^{(2)}$	...	$b_{s-1}^{(2)}$	$b_s^{(2)}$

Наборы параметров

$$b_n^{(1)} \text{ и } b_n^{(2)} \quad (n=1, \dots, s)$$

определяют методы порядка p и $p+1$, соответственно. Несколько примеров вложенных схем приведены ниже.

0	0				
$\frac{1}{4}$	$\frac{1}{4}$				
$\frac{4}{9}$	$\frac{4}{81}$	$\frac{32}{81}$			
$\frac{6}{7}$	$\frac{57}{98}$	$-\frac{432}{343}$	$\frac{1053}{686}$		
1	$\frac{1}{6}$	0	$\frac{27}{52}$	$\frac{49}{156}$	
	$\frac{1}{6}$	0	$\frac{27}{52}$	$\frac{49}{156}$	0
	43	0	243	343	$\frac{1}{12}$
	288		416	1872	

Пара Фельберга порядка 3(4)

0	0					
$\frac{1}{4}$	$\frac{1}{4}$					
$\frac{3}{8}$	$\frac{3}{32}$	$\frac{9}{32}$				
$\frac{12}{13}$	$\frac{1932}{2197}$	$\frac{7200}{2197}$	$\frac{7296}{2197}$			
1	$\frac{439}{216}$	-8	$\frac{3680}{513}$	$-\frac{845}{4104}$		
$\frac{1}{2}$	$-\frac{8}{27}$	2	$\frac{3544}{2565}$	$\frac{1859}{4104}$	$-\frac{11}{40}$	
	$\frac{25}{216}$	0	$\frac{1408}{2565}$	$\frac{2197}{4104}$	$-\frac{1}{5}$	0
	$\frac{16}{135}$	0	$\frac{6656}{12825}$	$\frac{28561}{56430}$	$-\frac{9}{50}$	$\frac{2}{55}$

Пара Фельберга порядка 4(5)

После того как приближённые решения

$$u_n^{(a)} \text{ и } \bar{u}_n^{(a)}$$

вычислены с некоторым шагом $h_{n,1}$, проверяется условие

$$|\bar{u}_n^{(a)} - u_n^{(a)}| \leq \varepsilon_p.$$

Если это условие не выполняется, то выбранный шаг $h_{n,1}$ не достаточно мал и следует повторить вычисления с другим шагом $h_{n,2}$. Если для вычисления $u^{(a)}$ используется метод с порядком аппроксимации p , тогда

$$|\bar{u}_n^{(a)} - u_n^{(a)}| \approx c h_{n,1}^{p+1}$$

и шаг $h_{n,2}$ выбирается из условия

$$h_{n,2} \leq h_{n,1} \left(\frac{\varepsilon_p}{|\bar{u}_n^{(a)} - u_n^{(a)}|} \right)^{\frac{1}{1+p}}.$$

Этот процесс повторяется до тех пор, пока приемлемое значение шага разностной сетки не будет найдено. Это значение, затем, может быть использовано в качестве начального шага для вычисления решения в следующей точке: $h_{n+1,1} = h_{n,2}$. При использовании схем предиктор-корректор (параграф 6.3.3), ошибка может быть оценена через разность двух решений полученных на этапах предиктор и корректор. Получим оценку ошибки в случае, когда предиктор (явная схема Адамса) и корректор (неявная схема Адамса) имеют одинаковый порядок аппроксимации. Тогда для метода порядка p , локальная ошибка записывается как

$$\bar{e}_{n+1} = u_{n+1}^{(e)} - \bar{u}_{n+1}^{(a)} = \alpha_p c h^p + O(h^{p+1}) \text{ (предиктор),}$$

$$e_{n+1} = u_{n+1}^{(e)} - u_{n+1}^{(a)} = \alpha_c c h^p + O(h^{p+1}) \text{ (корректор),}$$

где параметры α_c и α_p не зависят от h (значения этих параметров приведены в Таблице 6.6).

Таблица 6.6

Порядок аппроксимации, p	α_p	α_c
1	1/2	-1/2
2	5/12	-1/12
3	3/8	-1/24
4	251/720	-19/720

Теперь, мы можем выразить из этих двух формул ch^p (основную часть ошибки):

$$ch^p = \frac{u_{n+1}^{(a)} - \bar{u}_{n+1}^{(a)}}{\alpha_p - \alpha_c} + O(h^{p+1}).$$

Пренебрегая слагаемым порядка $p+1$, мы получим оценку локальной ошибки приближённого решения на этапе корректор:

$$u_{n+1}^{(e)} - u_{n+1}^{(a)} \approx \frac{\alpha_c}{\alpha_p - \alpha_c} (u_{n+1}^{(a)} - \bar{u}_{n+1}^{(a)})$$

Более того, выполняя локальную экстраполяцию, можно получить более точное решение чем $u_{n+1}^{(a)}$. Если мы запишем

$$u_{n+1}^{(e)} - (u_{n+1}^{(a)} + \alpha_c ch^p) = u_{n+1}^{(e)} - \hat{u}_{n+1}^{(a)} = O(h^{p+1}),$$

то приближённое решение

$$\hat{u}_{n+1}^{(a)} = \frac{\alpha_p u_{n+1}^{(a)} - \alpha_c \bar{u}_{n+1}^{(a)}}{\alpha_p - \alpha_c}$$

будет иметь ошибку порядка h^{p+1} .

В случае краевой задачи, мы можем оценить глобальную ошибку $\mathbf{e} = \mathbf{u}^{(e)} - \mathbf{u}^{(a)}$. Как и прежде, для оценки ошибки нам нужно два приближённых решения. В части 6.4 мы познакомились с разностными схемами для дифференциальных уравнений второго порядка. Эти схемы имели второй порядок аппроксимации, то есть основная часть ошибки была порядка h^2 . Пусть приближённое решение $\mathbf{u}^{(a)}$ получено при вычислении с шагом h , а решение $\mathbf{v}^{(a)}$ с шагом $h/2$. Тогда векторы ошибки этих решений записываются как

$$\mathbf{e}_1 = \mathbf{u}^{(e)} - \mathbf{u}^{(a)} = ch^2 + O(h^4)$$

$$\mathbf{e}_2 = \mathbf{u}^{(e)} - \mathbf{v}^{(a)} = c \left(\frac{1}{2} h \right)^2 + O(h^4)$$

Пренебрегая слагаемым порядка h^4 , можно выразить эти векторы через приближённые решения:

$$\mathbf{e}_1 \approx \frac{4}{3} (\mathbf{v}^{(a)} - \mathbf{u}^{(a)}),$$

$$\mathbf{e}_2 \approx \frac{1}{3} (\mathbf{v}^{(a)} - \mathbf{u}^{(a)}).$$

Если

$$\| \mathbf{e}_2 \| = \frac{1}{3} \| \mathbf{v}^{(a)} - \mathbf{u}^{(a)} \| \leq \varepsilon_p,$$

то $\mathbf{v}^{(a)}$ есть требуемое решение и вычисления заканчиваются. В противном случае, вычисляется решение с вдвое меньшим шагом разностной сетки и процесс оценки повторяется. Используя экстраполяцию можно построить решение

$$\hat{\mathbf{u}}^{(a)} = \frac{1}{3} (4\mathbf{v}^{(a)} - \mathbf{u}^{(a)}),$$

которое имеет четвёртый порядок точности.

ГЛАВА 7

Интерполяция и приближение функций

Список обозначений

$P_N(x)$	интерполяционный полином
$Q_N(x)$	тригонометрический интерполяционный полином
$s(x)$	интерполяционная сплайн функция
$\varphi_m(x)$	базисная функция
$T_n(x)$	полином Чебышева степени n
$x_{n,k}^*$	k -ый корень полинома Чебышева степени n
$\tilde{T}_n(x)$	нормированный полином Чебышева степени n

7.1. Интерполяция

Результаты экспериментов часто доступны в виде некоторого набора дискретных данных. С другой стороны, зависимость между различными параметрами какого-нибудь явления удобнее всего представлять в аналитической форме. Такая форма позволяет определить значения в промежуточных точках, вычислить интеграл или производную, или просто представить данные в виде гладкой функции.

Интерполяцией называется процедура определения функции, которая точно описывает заданный набор данных. Наиболее простой тип интерполяции состоит в построении полинома, проходящего через заданные точки. С вычислительной точки зрения полиномы имеют полезные свойства: производные и интегралы от них также являются полиномами и вычисление значения полинома в некоторой точке требует только умножений и сложений. Поэтому вначале, в параграфе 7.1.1, мы рассмотрим классические интерполяционные полиномы. В следующем параграфе мы обсудим применение тригонометрических полиномов для интерполяции периодических функций. В последних

двух параграфах рассматривается интерполяция кубическими сплайнами и кратко обсуждается один из методов двумерной интерполяции.

7.1.1. Интерполяционные полиномы Лагранжа и Ньютона.

Вначале, определим полином первой степени, проходящий через точки (x_0, y_0) и (x_1, y_1) . Линейный полином

$$P_1(x) = \frac{x - x_1}{x_0 - x_1} y_0 + \frac{x - x_0}{x_1 - x_0} y_1$$

имеет требуемое свойство, так как $P_1(x_0) = y_0$ и $P_1(x_1) = y_1$. Построим теперь полином степени не выше чем N , удовлетворяющий условиям

$$P_N(x_n) = y_n, \quad n = 0, \dots, N, \quad (7.1)$$

где значения x_n и y_n заданы. Такой полином определяется единственным образом и может быть представлен в следующем виде:

$$P_N(x) = \sum_{n=0}^N y_n l_n(x), \quad (7.2)$$

где сомножители $l_n(x)$ задаются как

$$l_n(x) = \prod_{\substack{m=0 \\ m \neq n}}^N \frac{x - x_m}{x_n - x_m}, \quad n = 0, \dots, N$$

Полином (7.2) называется интерполяционным полиномом Лагранжа степени N . Для вычисления значения интерполяционного полинома в некоторой точке x , не прибегая к явному вычислению коэффициентов полинома, применяется схема Невилла (Neville). На первом этапе, зададим значения

$$P_n^{(0)}(x_n) = y_n, \quad n = 0, \dots, N.$$

Далее, значения вспомогательных полиномов в точке x определяются согласно рекуррентному соотношению

$$P_n^{(k)}(x) = \frac{(x - x_n)P_{n+1}^{(k-1)}(x) - (x - x_{n+k})P_n^{(k-1)}(x)}{x_{n+k} - x_n}, \quad (7.3)$$

$$k = 1, \dots, N; n = 0, \dots, N - k$$

и, окончательно, мы получаем

$$P_0^{(N)}(x) = P_N(x).$$

Пример 7.1 (интерполяционный полином Лагранжа)

В Таблице 7.1 представлены значения y_n в соответствующих точках x_n и требуется провести гладкую кривую через эти точки.

Таблица 7.1

n	x_n	y_n
0	1	1
1	2	2
2	4	3
3	5	2

Определим сомножители $l_n(x)$, которые для данного набора данных записываются как

$$l_0(x) = \prod_{m=1}^3 \frac{x - x_m}{x_0 - x_m} = -\frac{1}{12}(x-2)(x-4)(x-5)$$

$$l_1(x) = \prod_{\substack{m=0 \\ m \neq 1}}^3 \frac{x - x_m}{x_1 - x_m} = \frac{1}{6}(x-1)(x-4)(x-5)$$

$$l_2(x) = \prod_{\substack{m=0 \\ m \neq 2}}^3 \frac{x - x_m}{x_2 - x_m} = -\frac{1}{6}(x-1)(x-2)(x-5)$$

$$l_3(x) = \prod_{m=0}^2 \frac{x - x_m}{x_3 - x_m} = \frac{1}{12}(x-1)(x-2)(x-4)$$

Тогда интерполяционный полином Лагранжа имеет вид

$$P_3(x) = \frac{1}{12}(3x-2)(x-4)(x-5) + \frac{1}{6}(11-2x)(x-1)(x-2).$$

На рисунке 7.1 показан интерполяционный полином Лагранжа, вместе с исходными данными из Таблицы 7.1. Предположим, что нам нужно вычислить значение $P_3(x)$ в промежуточной точке $x=3$, тогда схема Невилла (7.3) представляется в виде следующей таблицы:

k	$P_n^{(k)}(x), n = 0, \dots, 3-k$			
0	1	2	3	2
1	3	5/2	4	
2	8/3	3		
3	17/6			

Легко проверить, что $P_0^{(3)}(3) = P_N(3)$.

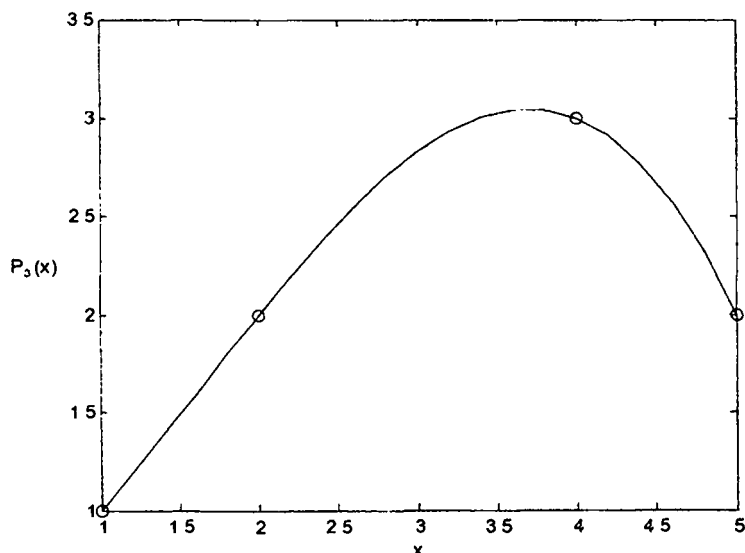


Рисунок 7.1. Гладкая кривая описывающая данные из Таблицы 7.1; о - данные; — - интерполяционный полином Лагранжа.

Представление интерполяционного полинома в форме (7.2) имеет простой вид и, поэтому, оно часто используется для теоретического анализа. Однако, с точки зрения практических вычислений, такое представление применимо только для небольших значений N . С увеличением N , сомножители $l_n(x)$ возрастают и начинают сильно осциллировать. Существует другое представление интерполяционного полинома, которое лучше подходит для вычислений, и оно основано на разделённых разностях. Пусть заданы различные точки $x_0, \dots, x_N$ и значения в этих точках $y_0, \dots, y_N$. Разделённые разности нулевого порядка есть просто значения y_n :

$$D_n^{(0)} = y_n, n = 0, \dots, N$$

Разделённые разности порядка k в точке x_n вычисляются по следующим рекуррентным формулам:

$$D_n^{(k)} = \frac{D_{n+1}^{(k-1)} - D_n^{(k-1)}}{x_{n+k} - x_n}, k = 1, \dots, N; n = 0, \dots, N - k$$

Тогда, используя введённые обозначения, интерполяционный полином, удовлетворяющий условию (7.1) записывается в виде

$$P_N(x) = D_0^{(0)} + \sum_{m=1}^N D_0^{(m)} \prod_{n=0}^{m-1} (x - x_n) \quad (7.4)$$

Полином (7.4) называется интерполяционным полиномом Ньютона. Его можно представить и в другой форме:

$$P_N(x) = \left\{ \dots \left(D_0^{(N)} (x - x_{N-1}) + D_0^{(N-1)} \right) (x - x_{N-2}) + \dots + D_0^{(1)} \right\} (x - x_0) + D_0^{(0)}$$

Используя это представление, значение интерполяционного полинома Ньютона в некоторой точке x может быть вычислено простым и эффективным способом. Этот алгоритм называется схемой Горнера (Horner) и задаётся следующими рекуррентными формулами:

$$a_N = D_0^{(N)}$$

$$a_k = a_{k+1}(x - x_k) + D_0^{(k)}, k = N - 1, \dots, 0'$$

Тогда требуемое значение полинома равно последнему элементу этой последовательности, то есть $P_N(x) = a_0$.

Пример 7.2 (интерполяционный полином Ньютона)

Рассмотрим данные, представленные в Таблице 7.1. Разделённые разности, вычисленные на основе этих данных, показаны в следующей таблице:

k	$D_n^{(k)}(x), n = 0, K, 3 - k$			
0	1	2	4	5
1	1	1/2	-1	
2	-1/6	-1/2		
3	-1/12			

Коэффициенты интерполяционного полинома Ньютона расположены в первом столбце этой таблицы. Тогда этот полином записывается в виде

$$P_3(x) = 1 + (x - 1) - \frac{1}{6}(x - 1)(x - 2) - \frac{1}{12}(x - 1)(x - 2)(x - 4)$$

7.1.2 Тригонометрические интерполяционные полиномы.

На практике мы часто сталкиваемся с периодическими функциями, которые имеют следующее свойство: $f(x+T)=f(x)$, где $T>0$ называется периодом функции. Например, функции, определённые на замкнутых плоских или пространственных кривых, могут рассматриваться как периодические функции. Полиномиальная интерполяция не подходит для таких зависимостей, так как алгебраические полиномы не являются периодическими функциями. Поэтому, мы обратимся к интерполяции с использованием тригонометрических полиномов. Пусть, для простоты, период T равен 2π . Предположим, что заданы значения $y_0, \dots, y_{2N}$ в

различных точках $x_0, \dots, x_{2N} \in [0, 2\pi)$. Тогда существует единственный тригонометрический полином $P_N(x)$ удовлетворяющий условию

$$Q_N(x_n) = y_n, n = 0, \dots, 2N.$$

В представлении Лагранжа этот полином имеет вид

$$Q_N(x) = \sum_{n=0}^{2N} y_n l_n(x), \quad (7.5)$$

где сомножители $l_n(x)$ задаются как

$$l_n(x) = \prod_{\substack{m=0 \\ m \neq n}}^{2N} \frac{\sin\left[\frac{1}{2}(x - x_m)\right]}{\sin\left[\frac{1}{2}(x_n - x_m)\right]}, n = 0, \dots, 2N.$$

Когда точки интерполяции расположены равномерно, представление (7.5) можно упростить. Для нечётного числа точек

$$x_n = \frac{2\pi n}{2N+1}, n = 0, \dots, 2N,$$

существует единственный тригонометрический полином

$$Q_N(x) = \frac{a_0}{2} + \sum_{k=1}^N (a_k \cos(kx) + b_k \sin(kx)) \quad (7.6)$$

удовлетворяющий условию

$$Q_N(x_n) = y_n, n = 0, \dots, 2N.$$

Коэффициенты этого полинома вычисляются следующим образом:

$$a_n = \frac{2}{2N+1} \sum_{m=0}^{2N} y_m \cos\left(\frac{2\pi mn}{2N+1}\right), n = 0, \dots, N$$

$$b_n = \frac{2}{2N+1} \sum_{m=0}^{2N} y_m \sin\left(\frac{2\pi mn}{2N+1}\right), n=1, \dots, N$$

Для чётного числа точек

$$x_n = \frac{\pi n}{N}, n=0, \dots, 2N-1,$$

существует единственный тригонометрический полином

$$Q_N(x) = \frac{1}{2}(a_0 + a_N \cos(Nx)) + \sum_{n=1}^{N-1} (a_n \cos(nx) + b_n \sin(nx)) \quad (7.7)$$

удовлетворяющий условию

$$Q_N(x_n) = y_n, n=0, \dots, 2N-1.$$

Коэффициенты этого полинома вычисляются следующим образом:

$$a_n = \frac{1}{N} \sum_{m=0}^{2N-1} y_m \cos\left(\frac{\pi mn}{N}\right), n=0, \dots, N$$

$$b_n = \frac{1}{N} \sum_{m=0}^{2N-1} y_m \sin\left(\frac{\pi mn}{N}\right), n=1, \dots, N-1$$

Тригонометрические интерполяционные полиномы (7.6) и (7.7) можно рассматривать как приближённый ряд Фурье, где интегралы, определяющие коэффициенты этого ряда, вычисляются по методу прямоугольников для равномерной сетки.

Для вычисления значения тригонометрического интерполяционного полинома в некоторой точке x следует использовать процедуру, аналогичную схеме Горнера для алгебраических полиномов. Только теперь рекуррентные формулы имеют следующий вид:

$$\begin{aligned} \alpha_{k-1} &= \alpha_k \cos(x) - \beta_k \sin(x) + a_{k-1}, \\ \beta_{k-1} &= \beta_k \cos(x) + \alpha_k \sin(x), \\ k &= N, \dots, 1, \end{aligned} \quad (7.8)$$

с начальными условиями $\alpha_N = a_N$ и $\beta_N = 0$. В итоге мы получим

$$\alpha_0 = \sum_{n=0}^N a_n \cos(nx).$$

Если вместо a_N подставить b_N , и задать начальное условие $\alpha_N = b_N$ и $\beta_N = 0$, то рекуррентные формулы (7.8) дают

$$\beta_0 = \sum_{n=1}^N b_n \sin(nx).$$

Следовательно, вычисление значения тригонометрического полинома в точке x требует только вычисления функций $\sin(x)$ и $\cos(x)$, $8N$ умножений и $6N$ сложений.

Пример 7.3 (тригонометрическая интерполяция)

Рассмотрим некоторые значения y_n , заданные в точках x_n , как показано в Таблице 7.2.

Таблица 7.2

n	0	1	2	3	4
x_n	0	$\pi/4$	$\pi/2$	π	$3\pi/2$
y_n	0	1	2	1	0.5

Предположим, что эти данные описываются некоторой периодической функцией. На рисунке 7.2 показаны тригонометрический полином $Q_4(x)$, определённый в (7.5), и интерполяционный полином Ньютона $P_4(x)$, построенный на основе данных из Таблицы 7.2.

Хорошо видно, что тригонометрический полином лучше представляет исходные данные, чем полином Ньютона. Более того, $Q_4(2\pi) = 0$, а $P_4(2\pi) \approx 19.2$, что демонстрирует непригодность полиномов для интерполяции периодических функций.

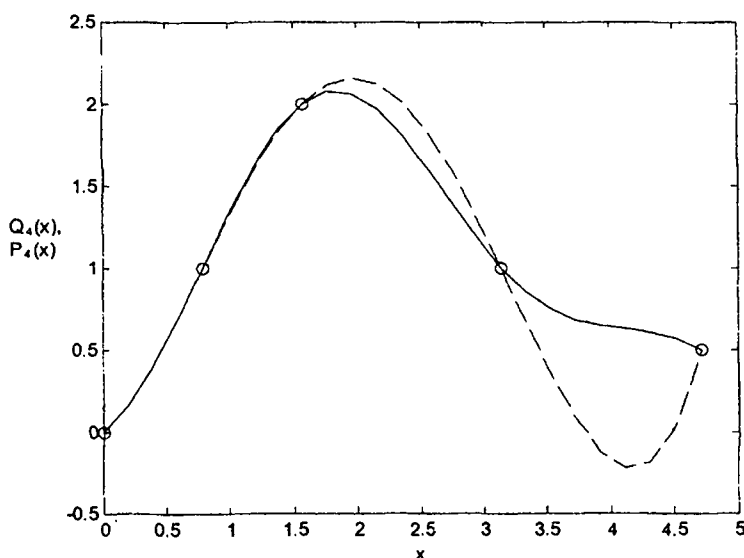


Рисунок 7.2. Гладкие кривые описывающие данные из Таблицы 7.2; о - данные; — - тригонометрический интерполяционный полином; — — интерполяционный полином Ньютона.

7.1.3 Сплайн интерполяция.

В предыдущих параграфах, мы использовали полиномы степени $N-1$ для того, чтобы представить набор данных из N значений. Однако, при больших значениях N , применение интерполяционных полиномов нецелесообразно. Поведение полиномов высоких степеней имеет осциллирующий характер, что может привести к неадекватному представлению данных.

Альтернативный подход состоит в использовании кусочно-полиномиальной интерполяции. Простейшим методом этого типа является кусочно-линейная интерполяция. Пусть значения $y_0, \dots, y_N$ заданы в различных точках $x_0, \dots, x_N$. Тогда эти данные могут быть представлены в виде непрерывной кусочно-линейной функции, которая принимает на каждом интервале $[x_n, x_{n+1}]$ следующую форму:

$$s_n(x) = \frac{1}{x_{n+1} - x_n} [y_n(x_{n+1} - x) + y_{n+1}(x - x_n)], x \in [x_n, x_{n+1}].$$

Такой подход имеет свой недостаток: интерполяционная функция, в общем, недифференцируема на концах интервалов, так как она имеет изгиб в этих точках. Однако, физические данные обычно представляются в виде гладких функций, поэтому интерполяционная функция должна быть непрерывно-дифференцируемой.

Наиболее популярный метод кусочно-полиномиальной интерполяции использует кубические полиномы на каждом интервале $[x_n, x_{n+1}]$ и носит название интерполяции кубическими сплайнами (или просто, сплайн интерполяции). Термин сплайн происходит от названия тонкой деревянной или металлической полосы, которая использовалась чертёжниками для того, чтобы провести гладкую кривую через заданные точки. Так как малое смещение w тонкой упругой полосы определяется уравнением четвёртого порядка $d^4w/dx^4=0$, то кубический полином действительно моделирует сплайн чертёжника.

Интерполяционная сплайн функция $s(x)$ есть объединение кубических полиномов $s_n(x)$: $s(x) = \bigcup s_n(x)$, определённых на каждом интервале $[x_n, x_{n+1}]$, и имеющих следующий вид:

$$s_n(x) = a_n + b_n(x - x_n) + c_n(x - x_n)^2 + d_n(x - x_n)^3 \quad (7.9)$$

Для того чтобы определить коэффициенты полиномов $s_n(x)$, задаются следующие условия:

- 1) $s_n(x_n) = y_n$, $n = 0, \dots, N-1$ и $s_{N-1}(x_N) = y_N$
- 2) непрерывность функции $s(x)$: $s_{n-1}(x_n) = s_n(x_n)$, $n = 0, \dots, N-1$
- 3) непрерывность первых производных функции $s(x)$:

$$\frac{ds_{n-1}}{dx}(x_n) = \frac{ds_n}{dx}(x_n), n = 1, \dots, N-1$$

- 4) непрерывность вторых производных функции $s(x)$:

$$\frac{d^2 s_{n-1}}{dx^2}(x_n) = \frac{d^2 s_n}{dx^2}(x_n), n = 1, \dots, N-1$$

- 5) некоторые граничные условия, например:

$$\frac{d^2 s_0}{dx^2}(x_0) = \frac{d^2 s_{N-1}}{dx^2}(x_N) = 0$$

Эти условия приводят к системе линейных уравнений для коэффициентов c_n кубического полинома (7.9):

$$c_0 = 0$$

$$\Delta x_{n-1} c_{n-1} + 2(\Delta x_n + x_{n-1}) c_n + \Delta x_n c_n =$$

$$\frac{3}{\Delta x_n} (y_{n+1} - y_n) - \frac{3}{\Delta x_{n-1}} (y_n - y_{n-1}),$$

$$c_N = 0$$

$$n = 1, \dots, N-1$$

где $\Delta x_n = x_{n+1} - x_n$. Матрица коэффициентов этой системы является трёхдиагональной и обладает свойством диагонального преобладания. Поэтому для решения этой системы можно использовать, например, метод прогонки (смотри параграф 2.6). После того как значения c_n вычислены, другие коэффициенты кубического полинома $s_n(x)$ определяются из следующих соотношений:

$$a_n = y_n, n = 0, \dots, N$$

$$b_n = \frac{1}{\Delta x_n} (y_{n+1} - y_n) - \frac{\Delta x_n}{3} (c_{n+1} + 2c_n)$$

$$d_n = \frac{1}{3\Delta x_n} (c_{n+1} - c_n),$$

$$n = 0, \dots, N-1$$

Пример 7.4 (интерполяция кубическими сплайнами)

В Таблице 7.3 представлены значения некоторой величины в 18-ти точках.

Таблица 7.3

n	x_n	y_n	n	x_n	y_n
0	0	1.3	9	5	2.2
1	0.4	1.5	10	6.1	2.3
2	1	1.8	11	7.1	2.2
3	1.2	2.1	12	8.3	1.9
4	1.7	2.6	13	9.6	1.4
5	2.1	2.7	14	10.4	0.9
6	3	2.4	15	10.7	0.7
7	3.5	2.1	16	11.1	0.6
8	3.8	2	17	11.7	0.3

Построим гладкую кривую проходящую через эти точки. Кубическая сплайн функция описывает эти данные достаточно хорошо, как показано на рисунке 7.3. Для сравнения, на этом рисунке также приведена кривая полученная с использованием интерполяционного полинома Лагранжа. В данном случае этот полином представляет собой полином 17-ой степени и он сильно осциллирует на конце интервала интерполяции.

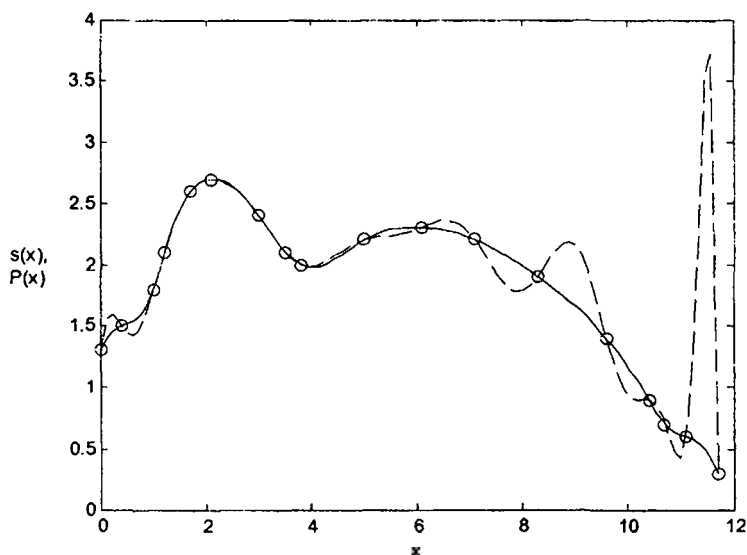


Рисунок 7.3. Гладкие кривые описывающие данные из Таблицы 7.3; о - данные; — — — интерполяционная сплайн функция; — — — интерполяционный полином Лагранжа.

7.1.4 Двумерная интерполяция.

Часто на практике возникает ситуация, когда необходимо построить функцию, представляющую данные, заданные в некоторых точках на плоскости. Предположим, что значения z_n , $n=1, \dots, N$ заданы в некоторых произвольно расположенных точках (x_n, y_n) , $n=1, \dots, N$ ($N \geq 3$).

Мы будем также предполагать, что эти точки попарно различны и не лежат на одной прямой. Интерполяционную функцию можно задать в виде

$$f(x, y) = \sum_{m=1}^N a_m b_m(x, y), \quad (7.10)$$

где $b_m(x, y)$ представляют собой некоторые базисные функции, а коэффициенты a_m определяются из условия интерполяции

$$f(x_n, y_n) = \sum_{m=1}^N a_m b_m(x_n, y_n) = z_n, \quad n=1, \dots, N \quad (7.11)$$

которое задаёт систему линейных уравнений для неизвестных a_m . Такой подход называется методом Харди (Hardy). Базисные функции в этом методе выбираются таким образом, чтобы функция $f(x, y)$ была достаточно гладкой и система уравнений (7.11) была бы как можно лучше обусловленной. Этим требованиям удовлетворяют, например, так называемые радиальные базисные функции

$$b_m(x, y) = b_m(r_m) = (r_m^2 + \alpha^2)^\beta \quad (7.12)$$

(на практике часто используется $\beta = 0.5$) и также

$$b_m(x, y) = b_m(r_m) = \ln(\alpha^2 + r_m^2),$$

$$b_m(x, y) = b_m(r_m) = (\alpha^2 + r_m^2)^k \ln(\alpha^2 + r_m^2), \quad k > 0$$

$$\text{здесь } r_m = \sqrt{(x - x_m)^2 + (y - y_m)^2}.$$

Параметр α следует выбирать не слишком большим, так как число обусловленности матрицы $C = \{c_{nm}\} = \{b_m(x_n, y_n)\}$ увеличивается с увеличением числа α . Величина этого параметра подбирается в каждом конкретном случае. В качестве ориентира, рекомендуется задавать

$$\alpha \sim \sqrt{\frac{1}{10} \max \left(\max_{n,m} |x_n - x_m|, \max_{n,m} |y_n - y_m| \right)},$$

что часто даёт хорошие результаты.

Пример 7.5 (двумерная интерполяция)

В Таблице 7.4 представлены значения некоторой величины в 5-ти точках на плоскости. Построим гладкую функцию проходящую через эти точки.

Таблица 7.4

n	x_n	y_n	z_n
1	0	0	0.6065
2	0	1	0.6065
3	1	1	0.6065
4	1	0	0.6065
5	0.5	0.5	1

На рисунке 7.4 показана интерполяционная функция (7.10) с базисными функциями, определёнными в (7.12).

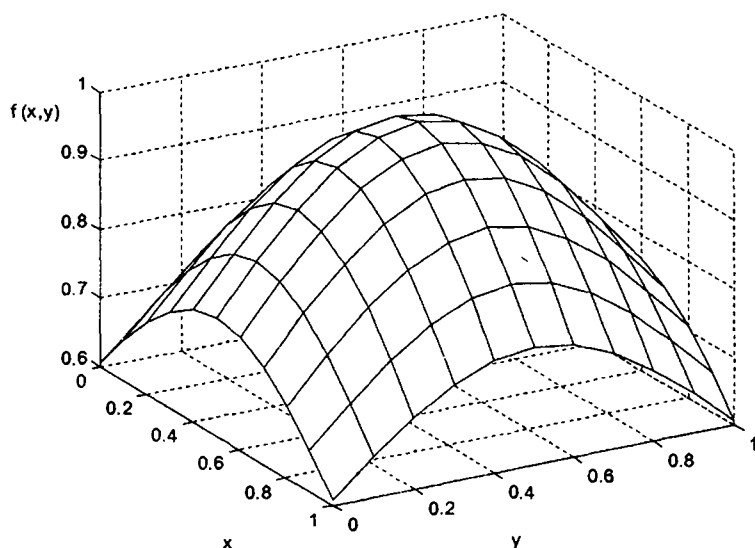


Рисунок 7.4. Двумерная интерполяционная функция, построенная методом Харди ($\alpha=0.8$, $\beta=0.5$).

На самом деле, значения x_n , приведённые в Таблице 7.4, вычислялись как $z_n = g(x_n, y_n) = \exp(-(0.5 - x_n)^2 - (0.5 - y_n)^2)$. Таким образом, мы пытаемся восстановить функцию, показанную на рисунке 7.5.

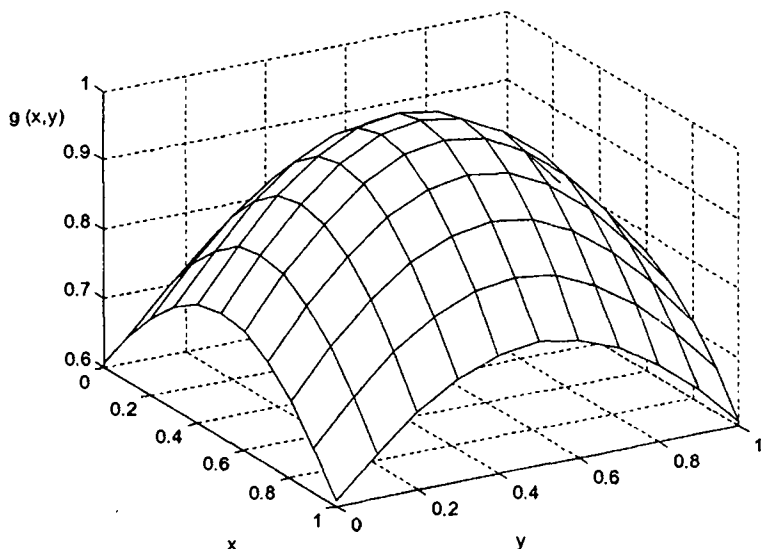


Рисунок 7.5. Функция $g(x,y)=\exp(-(0.5-x)^2-(0.5-y)^2)$.

Видно, что интерполяционная функция $f(x,y)$, показанная на рисунке 7.4, достаточно хорошо приближает «точную функцию» и глобальная относительная разница между $f(x,y)$ и $g(x,y)$ равна

$$\frac{\int_0^1 \int_0^1 |f(x,y) - g(x,y)| dx dy}{\int_0^1 \int_0^1 g(x,y) dx dy} \approx 0.01$$

7.2 Приближение функций и представление данных.

Теория приближения функций включает в себя два типа проблем. Первая проблема возникает, когда функция задана явно, но требуется найти другую функцию с которой, в данной ситуации, удобнее работать. Вторая проблема касается задачи определения функции,

принадлежащей некоторому классу, которая описывает заданный набор данных «наилучшим», в некотором смысле, образом. Мы начнём с рассмотрения этой проблемы.

7.2.1 Метод наименьших квадратов.

Пусть имеется некоторый набор экспериментальных данных, например, приведённый в Таблице 7.5, и мы хотим построить функцию, описывающую эти данные. На рисунке 7.6 приведено графическое представление данных из Таблицы 7.5.

Таблица 7.5

x_n	0	0.5	1	2	3	4	6	8
y_n	0.08	0.39	0.55	1.02	1.61	2.15	3.0	4.05

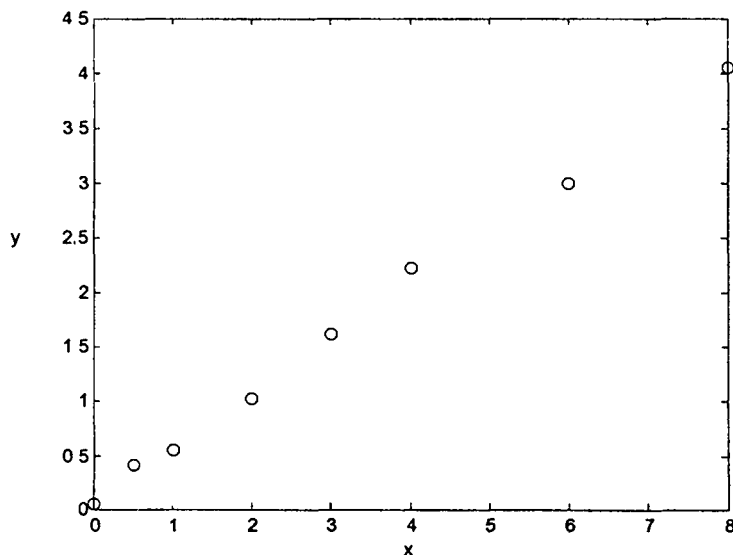


Рисунок 7.6. Графическое представление данных из Таблицы 7.5

Интерполяция требует чтобы интерполяционная функция принимала значения y_n в точках x_n для каждого $n=1, \dots, 8$. В данном случае,

требовать выполнения этого условия не имеет смысла, так как исходные данные не являются точными, а включают в себя погрешность измерений. Из графика видно, что, на самом деле, y зависит линейно от x . Поэтому, вместо интерполяции лучше использовать другой подход, а именно: построить линейную функцию, которая будет иметь «наименьшее» (в некотором смысле) отклонение от заданных значений, но, возможно, не проходить ни через одну точку.

Пусть $f_n = ax_n + b$ обозначает значение линейной функции в точке x_n и y_n есть данное значение в этой точке. Задачу определения линейной функции, которая описывает заданный набор данных, можно сформулировать следующим образом: найти такие значения коэффициентов a и b , чтобы ошибка

$$e(a, b) = \sum_{n=1}^N (y_n - f_n)^2 = \sum_{n=1}^N (y_n - (ax_n + b))^2$$

была минимальной. Условие минимума ошибки записывается в виде двух уравнений:

$$\frac{\partial}{\partial a} \sum_{n=1}^N (y_n - (ax_n + b))^2 = 2 \sum_{n=1}^N -x_n (y_n - ax_n - b) = 0$$

и

$$\frac{\partial}{\partial b} \sum_{n=1}^N (y_n - ax_n - b)^2 = -2 \sum_{n=1}^N (y_n - ax_n - b) = 0.$$

Преобразование этих выражений приводит к системе уравнений для коэффициентов a и b , имеющей вид

$$\begin{aligned} a \sum_{n=1}^N x_n^2 + b \sum_{n=1}^N x_n &= \sum_{n=1}^N x_n y_n \\ a \sum_{n=1}^N x_n + bN &= \sum_{n=1}^N y_n \end{aligned} \quad (7.13)$$

Решая эту систему, получим явные выражения для коэффициентов a и b :

$$a = \frac{Ng_1 - c_2 g_2}{Nc_1 - c_2^2}, \quad b = \frac{c_1 g_2 - c_2 g_1}{Nc_1 - c_2^2},$$

где

$$c_1 = \sum_{n=1}^N x_n^2, c_2 = \sum_{n=1}^N x_n$$

$$g_1 = \sum_{n=1}^N x_n y_n, g_2 = \sum_{n=1}^N y_n$$

Рассмотренный нами подход называется методом наименьших квадратов.

Пример 7.6 (построение линейной зависимости методом наименьших квадратов)

Рассмотрим данные приведённые в Таблице 7.5. В этом случае, решение системы уравнений (7.13) даёт $a \approx 0.4937$ и $b \approx 0.0943$. График линейной функции $f=ax+b$ и исходные данные показаны на рисунке 7.7. При этом ошибка полученного приближения $e \approx 1.9 \cdot 10^{-2}$.

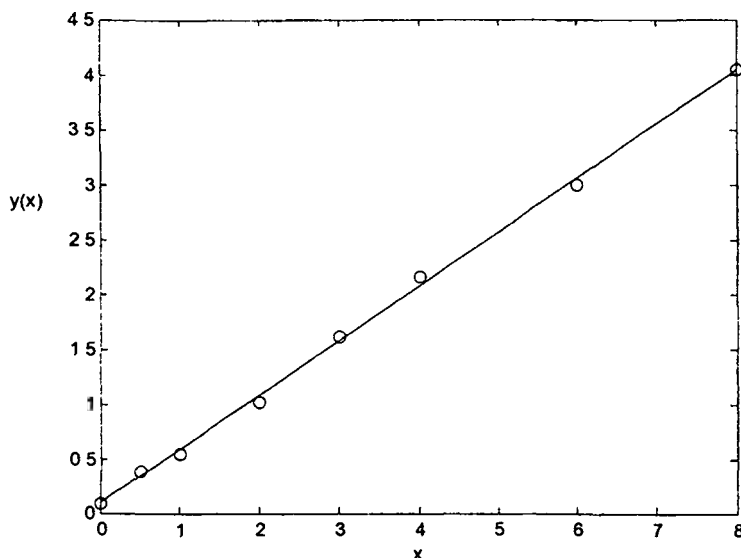


Рисунок 7.7. Линейная функция, приближающая данные из Таблицы 7.5; о - данные; — - приближение по методу наименьших квадратов.

$$e = \sum_{n=1}^N (y_n - f(x_n, a, b))^2,$$

но в данном случае, лучше записать ошибку в следующей форме:

$$e^{(1)} = \sum_{n=1}^N (\ln(y_n) - \ln(a) + bx_n)^2.$$

Теперь мы можем минимизировать ошибку $e^{(1)}$ относительно параметров $\ln(a)$ и b , что даёт линейную систему уравнений для этих параметров, аналогичную системе (7.13).

Пример 7.7 (построение полинома методом наименьших квадратов)

Рассмотрим данные приведённые в Таблице 7.6.

Таблица 7.6

n	x_n	y_n	n	x_n	y_n
1	0	0	7	1.2	0.9
2	0.2	0.32	8	1.4	0.92
3	0.4	0.58	9	1.6	0.96
4	0.6	0.68	10	1.8	0.99
5	0.8	0.8	11	2.0	1.03
6	1.0	0.88			

Предположим, что эти данные можно приближённо описать полиномом третьей степени. Тогда решение системы (7.14) при $M=3$, $N=11$ даёт следующие значения коэффициентов этого полинома:

$$a_0 \approx 0.009, \quad a_1 \approx 1.7738, \quad a_2 \approx -1.1926, \quad a_3 \approx 0.2817$$

На рисунке 7.8 показан график полинома $P_3(x) = 0.009 + 1.7738x - 1.1926x^2 + 0.2817x^3$ вместе с исходными данными из Таблицы 7.6.

Ошибка полученного приближения равна

$$\sum_{i=1}^{11} (y_n - P_3(x_n))^2 \approx 2.5 \cdot 10^{-3}.$$

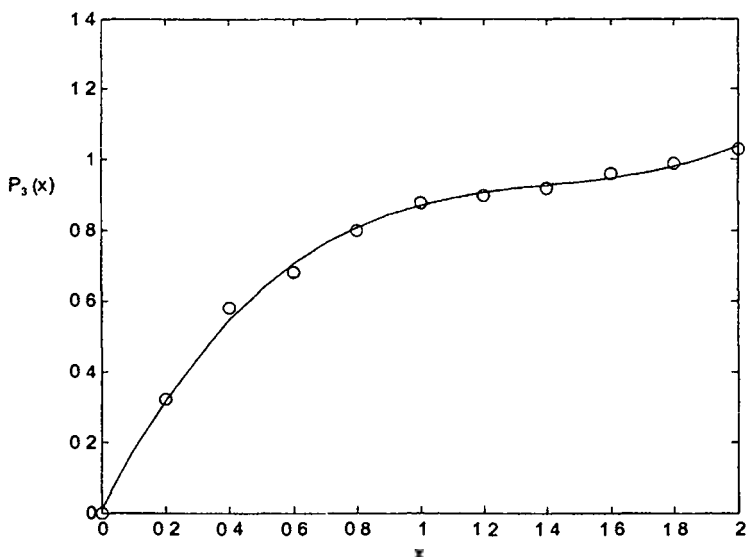


Рисунок 7.8. Кубический полином, полученный методом наименьших квадратов, для данных из Таблицы 7.6; о - данные; — - приближение по методу наименьших квадратов.

7.2.2 Приближение функции набором ортогональных функций

Теперь мы обратим наше внимание на задачу приближения (аппроксимации) функций. В предыдущем параграфе мы рассмотрели метод наименьших квадратов для приближения дискретных данных. Применим этот подход для построения полинома, который аппроксимирует непрерывную функцию $f(x)$, определённую на интервале $[a, b]$. Только в этом случае, суммирование следует заменить на определённый интеграл. Следуя процедуре метода наименьших квадратов, коэффициенты полинома

$$P_M(x) = \sum_{m=0}^M a_m x^m$$

степени M выбираются таким образом, чтобы ошибка

$$e(a_0, \dots, a_M) = \int_a^b (f(x) - P_M(x))^2 dx$$

была минимальной. Как и прежде, условия минимума ошибки имеют вид

$$\frac{\partial e}{\partial a_m} = 0, m = 0, \dots, M,$$

то есть

$$\int_a^b (f(x) - P_M(x)) dx = 0$$

$$\int_a^b x(f(x) - P_M(x)) dx = 0$$

.....

$$\int_a^b x^M (...) dx = 0$$

Дальнейшее преобразование этих выражений даёт систему линейных уравнений для коэффициентов a_m :

$$\sum_{m=0}^M a_m \int_a^b x^{n+m} dx = \int_a^b x^n f(x) dx \quad (7.15)$$

$$n = 0, \dots, M$$

Эта система уравнений всегда имеет единственное решение. Элемент с номером (n, m) матрицы коэффициентов системы (7.15) определяется как

$$\int_a^b x^{n+m} dx = \frac{b^{n+m+1} - a^{n+m+1}}{n+m+1}.$$

Матрицы такого типа называются матрицами Гильберта и они очень плохо обусловлены, что приводит к значительному искажению результатов решения (смотри параграф 2.5). В дополнение, рассмотренный нами подход не эффективен с вычислительной точки зрения, так как

информация полученная при построении полинома степени N не может быть использована при построении полинома более высокой степени.

Эти затруднения можно преодолеть, если выбрать аппроксимирующую функцию специальным образом, а именно:

$$g(x) = \sum_{m=0}^M a_m \varphi_m(x), \quad (7.16)$$

где $\varphi_m(x)$ ($m=0, \dots, M$) представляет собой набор базисных функций или полиномов. Далее мы покажем, что при определённом выборе этих базисных функций, можно избежать решения системы уравнений. Коэффициенты a_m ($m=0, \dots, M$) выбираются из условия минимума ошибки

$$e(a_0, \dots, a_M) = \int_a^b \rho(x)(f(x) - g(x))^2 dx,$$

где $\rho(x)$ называется весовой функцией. Эта функция должна быть неотрицательной интегрируемой на $[a, b]$ функцией, и она также не должна обращаться в ноль на любом интервале из отрезка $[a, b]$. Условия минимума ошибки записываются в виде

$$\frac{\partial e}{\partial a_m} = -2 \int_a^b \rho(x) \varphi_m(x) \left(f(x) - \sum_{m=0}^M a_m \varphi_m(x) \right) dx = 0, \\ m = 0, \dots, M$$

Преобразование этих выражений даёт систему линейных уравнений для коэффициентов a_m :

$$\sum_{m=0}^M a_m \int_a^b \rho(x) \varphi_m(x) \varphi_n(x) dx = \int_a^b \rho(x) f(x) \varphi_n(x) dx \\ n = 0, \dots, M \quad (7.17)$$

Пусть функции $\varphi_m(x)$ удовлетворяют следующему условию:

$$\int_a^b \rho(x) \varphi_n(x) \varphi_m(x) dx = \begin{cases} 0, & \text{if } n \neq m \\ \alpha_n > 0, & \text{if } n = m \end{cases} \quad (7.18)$$

Тогда, только один интеграл в левой части уравнений (7.17) будет отличен от нуля и решение этой системы имеет простой вид:

$$a_n = \frac{1}{\alpha_n} \int_a^b \rho(x) f(x) \varphi_n(x) dx.$$

Функции удовлетворяющие условию (7.18) называются ортогональными на отрезке $[a, b]$ с весом $\rho(x)$ функциями. Они имеют очень полезное свойство: улучшение аппроксимации, посредством включения дополнительного слагаемого $a_{M+1} \varphi_{M+1}(x)$, не влияет на уже вычисленные коэффициенты $a_0, \dots, a_M$. Замена переменных

$$x = \frac{b-a}{d-c} \bar{x} - \frac{cb-ad}{d-c}, dx = \frac{b-a}{d-c} d\bar{x} \quad (7.19)$$

где $x \in [a, b]$ и $\bar{x} \in [c, d]$, позволяет использовать формулы (7.16)-(7.18) для произвольного конечного интервала аппроксимации.

Ортогональные функции (полиномы) играют важную роль в математической физике и механике, поэтому приближение (7.16) может быть полезным в различных ситуациях. Далее мы рассмотрим некоторые примеры применения ортогональных функций и полиномов к задаче аппроксимации функций. Вначале мы обсудим разложение по полиномам Чебышева $T_n(x)$, которое имеет вид

$$g(x) = \sum_{m=0}^M a_m T_m(x) \quad (7.20)$$

Полином (7.20) замечателен тем, что он близок к идеальному полиному $P_{opt}(x)$ степени M , который имеет наименьшее отклонение от заданной функции на отрезке $[a, b]$, то есть

$$\max_{x \in [a, b]} |f(x) - P_{opt}(x)| \leq \max_{x \in [a, b]} |f(x) - P(x)|,$$

где $P(x)$ - любой полином степени M . Полиномы Чебышева $T_n(x)$ ортогональны на отрезке $[1, -1]$ с весом $\rho(x) = (1-x^2)^{-1/2}$ и константа ортогональности из выражения (7.18) равна $\alpha_n = \pi/2$. Для $x \in [-1, 1]$, они определяются как $T_n(x) = \cos(n \cdot \arccos(x))$ для каждого значения $n \geq 0$ или

$$T_0(x) = 1, T_1(x) = x$$

и

$$T_{n+1}(x) = 2xT_n(x) - T_{n-1}(x) \text{ для } n \geq 1.$$

Несколько первых полиномов $T_n(x)$ имеют следующий вид:

$$T_0(x) = 1$$

$$T_1(x) = x$$

$$T_2(x) = 2x^2 - 1$$

$$T_3(x) = 4x^3 - 3x$$

$$T_4(x) = 8x^4 - 8x^2 + 1$$

$$T_5(x) = 16x^5 - 20x^3 + 5x$$

$$T_6(x) = 32x^6 - 48x^4 + 18x^2 - 1$$

$$T_7(x) = 64x^7 - 112x^5 + 56x^3 - 7x$$

$$T_8(x) = 128x^8 - 256x^6 + 160x^4 - 32x^2 + 1$$

Полином Чебышева $T_N(x)$ степени $N \geq 1$ имеет N простых нулей на отрезке $[1, -1]$ в точках

$$x_{N,n}^* = \cos\left(\frac{2n-1}{2N} \pi\right), \quad n = 1, \dots, N \quad (7.21)$$

Ошибка приближения (7.20) равна, в основном, отброшенному слагаемому $a_{M+1}T_{M+1}(x)$. Так как полином $T_{M+1}(x)$ осциллирует на отрезке $[1, -1]$ с амплитудой не превышающей единицу, то максимальная ошибка приближения может быть оценена величиной $|a_{M+1}|$.

Пример 7.8 (аппроксимация функции с использованием полиномов Чебышева)

Пусть нам необходимо аппроксимировать функцию $f(x) = \ln(1+x)$ на отрезке $[0, 1]$. Полиномы Чебышева были определены на отрезке $[1, -1]$,

поэтому применяя замену переменных (7.19) с $a=-1$, $b=1$, $c=0$, и $d=1$ мы получим смещённые полиномы Чебышева, определённые на отрезке $[0,1]$. Тогда аппроксимирующий полином (7.20) принимает следующий вид (для демонстрации мы выберем бесконечный предел интегрирования):

$$\ln(1+x) \approx g(x) = \sum_{m=0}^{\infty} a_m T_m(2\bar{x}-1), \bar{x} \in [0,1]$$

Первые несколько коэффициентов этого ряда, с точностью до шести знаков, равны

$$a_0 \approx 0.376452$$

$$a_1 \approx 0.343145$$

$$a_2 \approx -0.029437$$

$$a_3 \approx 0.003367$$

$$a_4 \approx -0.000433$$

$$a_5 \approx 0.000059$$

$$a_6 \approx 0.000008$$

Например, если нам достаточно аппроксимации с ошибкой

$$\max_{x \in [0,1]} |\ln(1+x) - g(x)| \leq 10^{-3},$$

тогда достаточно оставить четыре слагаемых в приведённом выше разложении.

Тригонометрические функции используются для аппроксимации периодических функций, которые имеют следующее свойство: $f(x+T)=f(x)$, где $T>0$ называется периодом функции. Пусть, для простоты, период T равен 2π и интервал аппроксимации $[-\pi, \pi]$. Рассмотрим следующий набор функций:

$$\varphi_0(x) = \frac{1}{\sqrt{2\pi}}$$

$$\varphi_m(x) = \frac{1}{\sqrt{\pi}} \cos(mx), m = 1, \dots, M$$

$$\varphi_{M+m}(x) = \frac{1}{\sqrt{\pi}} \sin(mx), m = 1, \dots, M-1$$

Эти функции ортогональны на отрезке $[-\pi, \pi]$ с весом $\rho(x)=1$, и константа α_n из (7.18) равна единице. Для непрерывной на отрезке $[-\pi, \pi]$ функции $f(x)$, аппроксимирующая функция (7.16) задаётся в виде

$$g(x) = \frac{a_0}{2\pi} + \frac{1}{\pi} \sum_{m=1}^M a_m \cos(mx) + \frac{1}{\pi} \sum_{m=1}^{M-1} b_m \sin(mx)$$

где

$$a_m = \int_{-\pi}^{\pi} f(x) \cos(mx) dx, m = 0, \dots, M$$

$$b_m = \int_{-\pi}^{\pi} f(x) \sin(mx) dx, m = 1, \dots, M-1$$

Так как амплитуда колебаний функций $\cos(mx)$ и $\sin(mx)$ не превышает единицу, то максимальную абсолютную ошибку аппроксимации $\max|f(x)-g(x)|$ можно оценить величиной

$$\frac{1}{\pi} (|a_M| + |b_{M-1}|)$$

Пример 7.9 (аппроксимация периодической функции)

Построим аппроксимацию для функции

$$f(x) = f_0(1 - |x/d|), x \in [-d, d]$$

Используя замену переменных, можно легко найти, что

$$a_0 = \frac{f_0 \pi}{d} \int_{-d}^d \left(1 - \left|\frac{x}{d}\right|\right) dx = \pi f_0,$$

$$a_m = \frac{f_0 \pi}{d} \int_{-d}^d \left(1 - \left|\frac{x}{d}\right|\right) \cos\left(\frac{\pi m x}{d}\right) dx = \frac{2f_0}{\pi m^2} [1 - (-1)^m],$$

$$m = 1, \dots, M$$

Так как исходная функция чётная, то

$$b_m = \frac{f_0 \pi}{d} \int_{-d}^d \left(1 - \left|\frac{x}{d}\right|\right) \sin\left(\frac{\pi m x}{d}\right) dx = 0.$$

Окончательно, аппроксимирующая функция имеет вид

$$g(x) = \frac{f_0}{2} + \frac{2f_0}{\pi^2} \sum_{m=1}^M \frac{1 - (-1)^m}{m^2} \cos\left(\frac{\pi m x}{d}\right) = \frac{f_0}{2} + \frac{4f_0}{\pi^2} \sum_{n=0}^N \frac{1}{(2n+1)^2} \cos\left[\frac{\pi(2n+1)x}{d}\right]$$

В дополнение, мы можем оценить число членов ряда N , необходимое для получения аппроксимации с заданной точностью ε_p . В нашем случае

$$\frac{1}{\pi} |a_{M+1}| = \frac{4f_0}{\pi^2 (2N+3)^2} \leq \varepsilon_p$$

и, следовательно,

$$N = \left\lceil \frac{1}{\pi} \sqrt{f_0 / \varepsilon_p} - 1.5 \right\rceil + 1.$$

Графики аппроксимирующей функции с $M=3$ и функции $f(x)=1-|x/d|$ показаны на рисунке 7.9.

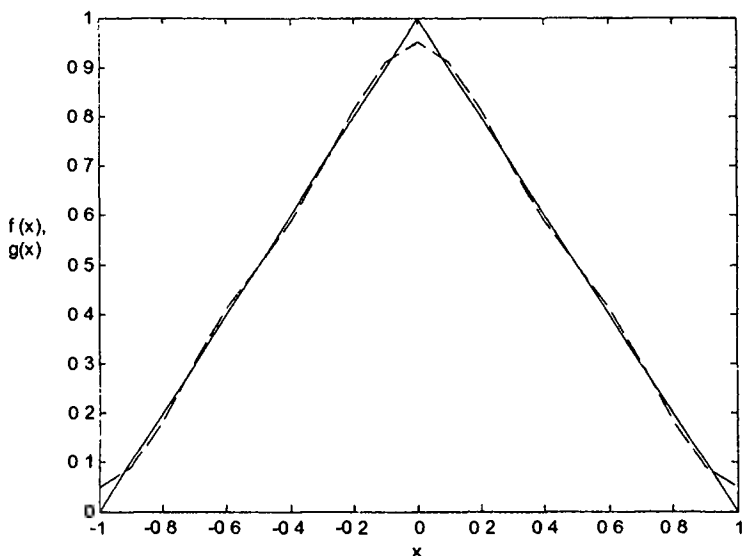


Рисунок 7.9. Тригонометрическая аппроксимация для функции $f(x)=1-|x/d|$ ($M=3$, $f_0=1$, $d=1$); — — — $f(x)$; — — — $g(x)$.

7.2.3 Использование интерполяционных полиномов для приближения функций.

Интерполяционные полиномы, рассмотренные в параграфе 7.1.1, также можно применить для приближения непрерывной функции $f(x)$ на некотором интервале $[a, b]$. В этом случае точки $x_0, \dots, x_N \in [a, b]$ задаются произвольно, а $y_n = f(x_n)$. Предположим, что функция $f(x)$ определена на интервале $[-1, 1]$. Если мы построим интерполяционный полином, который приближает функцию $f(x)$, то ошибку этого приближения можно представить в следующем виде:

$$f(x) - P_N(x) = \frac{1}{(N+1)!} \frac{d^{N+1} f}{dx^{N+1}}(z(x)) \prod_{n=0}^N (x - x_n),$$

$$x \in [-1, 1], z(x) \in [-1, 1],$$

где $P_N(x)$ обозначает интерполяционный полином Лагранжа. Минимизация ошибки сводится к минимизации величины $(x-x_0)(x-x_1)\dots(x-x_N)$. Это можно осуществить, располагая узлы $x_0, \dots, x_N$ специальным образом. Прежде чем обратиться к этому вопросу, нам необходимо рассмотреть некоторые дополнительные свойства полиномов Чебышева.

Нормированный полином Чебышева $T_n(x)$ образуется из полинома Чебышева $T_n(x)$ делением на 2^{n-1} , где $n \geq 1$ (коэффициент при старшей степени полинома $T_n(x)$ равен единице). Тогда

$$T_0^*(x) = 1,$$

$$T_n^*(x) = 2^{1-n} T_n(x), \text{ при } n \geq 1.$$

При этом нули полинома $T_n^*(x)$ совпадают с нулями полинома $T_n(x)$, и они задаются выражением (7.21). Полиномы $T_n^*(x)$, $n \geq 1$, обладают уникальным свойством: они имеют наименьшее отклонение от нуля на отрезке $[-1, 1]$ среди всех полиномов степени n . Так как $(x-x_0)(x-x_1)\dots(x-x_N)$ представляет собой нормированный полином степени $N+1$, то минимальное отклонение этой величины от нуля достигается тогда, когда

$$\prod_{n=0}^N (x - x_n) = T_{N+1}^*(x).$$

Следовательно, узлы x_n , $n=0, \dots, N$, должны совпадать с нулями полинома $T_{N+1}(x)$, то есть $x_n = x_{N+1, n+1}$. Этот метод выбора узлов интерполяционного полинома можно применять и в случае произвольного конечного интервала $[a, b]$, используя замену переменных

$$\begin{aligned}\bar{x}_n &= \frac{1}{2}[(b-a)x_{N+1, n+1}^* + (a+b)], \\ \bar{x}_n &\in [a, b]\end{aligned}\quad (7.22)$$

Пример 7.10 (приближение функции полиномом Лагранжа)

Рассмотрим специальную функцию

$$Si(x) = \int_0^x \frac{\sin(t)}{t} dt \quad \text{для } x \in [0, 10].$$

Вычисление значений этой функции требует значительных вычислительных затрат. Для того чтобы уменьшить эти затраты, можно построить интерполяционный полином, аппроксимирующий данную функцию. Зададим, например, семь узлов интерполирования, определяемых по формуле (7.22) ($N=6$, $a=0$, $b=10$):

$$\bar{x}_0 \approx 0.096074$$

$$\bar{x}_1 \approx 0.842652$$

$$\bar{x}_2 \approx 2.222149$$

$$\bar{x}_3 \approx 4.024548$$

$$\bar{x}_4 \approx 5.975452$$

$$\bar{x}_5 \approx 7.777851$$

$$\bar{x}_6 \approx 9.157348$$

Теперь мы можем построить интерполяционный полином Лагранжа (7.2), принимающий в этих точках значения $y_n = Si(\bar{x}_n)$. На рисунке 7.10 показаны графики полученного интерполяционного полинома и функции $Si(x)$. Легко видеть, что этот полином даёт хорошее приближение функции $Si(x)$ на отрезке $x \in [0, 9]$.

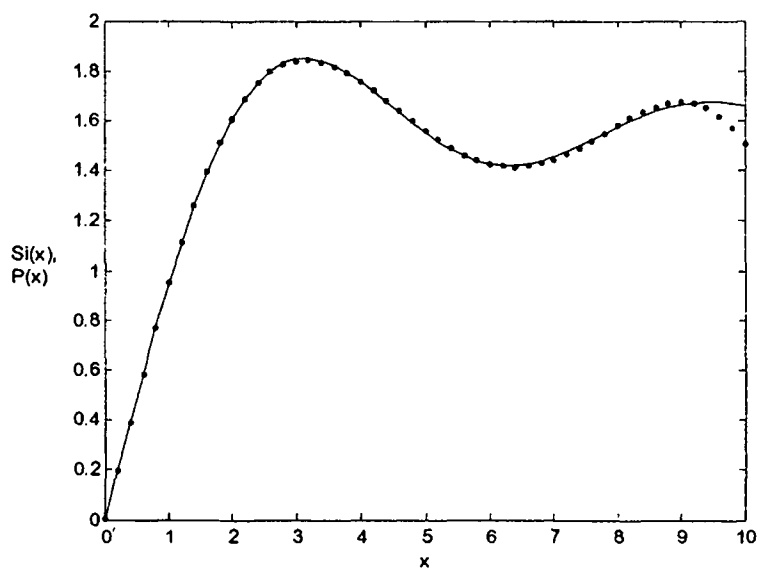


Рисунок 7.10. Аппроксимация функции $Si(x)$ интерполяционным полиномом Лагранжа с оптимальными узлами интерполяции; — - $Si(x)$; • - интерполяционный полином.

ПРИЛОЖЕНИЕ А

Узлы и веса некоторых квадратурных формул Гаусса-Кристоффеля

Это приложение содержит узлы и веса квадратурных формул Гаусса-Кристоффеля (5.14) для интегралов вида (5.13) с весовыми функциями приведёнными в (5.17).7

$$1) \rho(x) = 1, c = -1, d = 1$$

Соответствующие ортогональные полиномы:
полиномы Лежандра $P_m(x)$, $P_m(1) = 1$

Таблица А1.

M	Узлы z_m , $m = 1, \dots, M$	Веса w_m , $m = 1, \dots, M$	C(M)
2	-0.577350 0.577350	1.000000 1.000000	7.4e-03
3	-0.774597 0.000000 0.774597	0.555556 0.888889 0.555556	6.3e-05
4	-0.861136 -0.339981 0.339981 0.861136	0.347855 0.652145 0.652145 0.347855	2.9e-07
5	-0.906180 -0.538469 0.000000 0.538469 0.906180	0.236927 0.478629 0.568889 0.478629 0.236927	8.1e-10

$$2) \rho(x) = x^k, c = 0, d = 1$$

Соответствующие ортогональные полиномы:

$$q_M(x) = \sqrt{k + 2M + 1} P_M^{(k,0)}(1 - 2x),$$

где $P_M^{(k,0)}(x)$ - полиномы Якоби.

Таблица А2.

k=1

M	Узлы z_m , $m = 1, \dots, M$	Веса w_m , $m = 1, \dots, M$	C(M)
1	0.666667	0.500000	1.4e-02
2	0.355051 0.844949	0.181959 0.318041	6.9e-05
3	0.212334 0.590533 0.911412	0.069827 0.229241 0.200932	1.4e-07
4	0.139760 0.416410 0.723157 0.942896	0.031181 0.129848 0.203465 0.135507	1.6e-10
5	0.098535 0.304536 0.562025 0.801987 0.960190	0.015748 0.073909 0.146387 0.167175 0.096782	1.1e-13

Таблица А3.

k=2			
M	Узлы z_m , $m = 1, \dots, M$	Веса w_m , $m = 1, \dots, M$	C(M)
1	0.750000	0.333333	6.3e-03
2	0.455848 0.877485	0.101786 0.232547	2.6e-05
3	0.294998 0.652996 0.927006	0.029951 0.146246 0.157136	4.9e-08
4	0.204149 0.482953 0.761399 0.951499	0.010352 0.068634 0.143459 0.110888	5.1e-11
5	0.148946 0.365667 0.610114 0.826520 0.965421	0.004114 0.032056 0.089200 0.126199 0.081765	3.4e-14

Таблица А4.

k=3

M	Узлы z_m , $m = 1, \dots, M$	Веса w_m , $m = 1, \dots, M$	C(M)
1	0.800000	0.250000	3.3e-03
2	0.529858 0.898713	0.066905 0.183095	1.2e-05
3	0.363265 0.698811 0.937924	0.016479 0.104600 0.128921	2.0e-08
4	0.261478 0.535846 0.790283 0.957847	0.004658 0.042542 0.109004 0.093796	1.9e-11
5	0.196212 0.417100 0.648570 0.845605 0.969436	0.001521 0.016957 0.060445 0.100317 0.070761	1.2e-14

Таблица А5.

k=4

M	Узлы z_m , $m = 1, \dots, M$	Веса w_m , $m = 1, \dots, M$	C(M)
1	0.833333	0.200000	2.0e-03
2	0.586337 0.913663	0.049082 0.150918	5.9e-06
3	0.420113 0.733889 0.945998	0.010469 0.080277 0.109254	8.8e-09
4	0.312135 0.578916 0.812892 0.962724	0.002516 0.029169 0.087068 0.081247	7.8e-12
5	0.239792 0.460937 0.680059 0.860886 0.972614	0.000697 0.010211 0.044024 0.082713 0.062355	4.6e-15

Таблица А6.

k=5

M	Узлы z_m , $m = 1, \dots, M$	Веса w_m , $m = 1, \dots, M$	C(M)
1	0.857143	0.166667	1.3e-03
2	0.630792 0.924764	0.038338 0.128329	3.2e-06
3	0.467983 0.761624 0.952211	0.007297 0.064597 0.094773	4.3e-09
4	0.356894 0.614669 0.831079 0.966589	0.001534 0.021428 0.072056 0.071647	3.5e-12
5	0.279693 0.498710 0.706334 0.873403 0.975194	0.000370 0.006730 0.033768 0.070071 0.055728	4.6e-15

$$3) \rho(x) = \sqrt{1-x}, \quad c = 0, \quad d = 1$$

Соответствующие ортогональные полиномы:

$$\frac{1}{\sqrt{1-x}} P_{2M+1}(\sqrt{1-x}), \quad P_{2M+1}(1) = 1$$

Таблица А7.

М	Узлы z_m , $m = 1, \dots, M$	Веса w_m , $m = 1, \dots, M$	C(M)
1	0.400001	0.666664	2.3e-02
2	0.710051 0.178839	0.277555 0.389108	1.2e-04
3	0.835289 0.450131 0.099195	0.125782 0.307601 0.233279	2.6e-07
4	0.894859 0.623776 0.301052 0.062666	0.065680 0.196095 0.252527 0.152361	2.9e-10
5	0.927347 0.730539 0.466878 0.213121 0.043069	0.038186 0.125672 0.198630 0.197632 0.106540	2.0e-13

$$4) \rho(x) = \frac{1}{\sqrt{1-x}}, \quad c = 0, \quad d = 1$$

Соответствующие ортогональные полиномы:

$$P_{2m}(\sqrt{1-x}), \quad P_{2m}(1) = 1$$

Таблица А8.

M	Узлы z_m , $m = 1, \dots, M$	Веса w_m , $m = 1, \dots, M$	C(M)
1	0.666667	2.000000	8.9e-02
2	0.884412 0.223599	1.304290 0.695708	4.8e-04
3	0.943061 0.562802 0.130501	0.935828 0.721524 0.342648	1.0e-06
4	0.966352 0.723816 0.355323 0.077843	0.725368 0.627414 0.444762 0.202458	1.1e-09
5	0.977837 0.812169 0.538402 0.251666 0.051505	0.591048 0.538534 0.438172 0.298902 0.133342	8.1e-13

$$5) \rho(x) = \frac{1}{\sqrt{1-x^2}}, \quad c = -1, \quad d = 1$$

Соответствующие ортогональные полиномы:
полиномы Чебышева первого рода

$$T_m(x), \quad T_m(1) = \frac{1}{2^{m-1}}$$

Узлы:

$$z_m = \cos\left(\frac{\pi(2m-1)}{2M}\right), \quad m = 1, \dots, M$$

Веса:

$$w_m = \frac{\pi}{M}, \quad m = 1, \dots, M$$

M	C(M)
1	7.9e-01
2	1.6e-02
3	1.4e-04
4	6.1e-07
5	1.7e-09

$$6) \rho(x) = \sqrt{1-x^2}, \quad c = -1, \quad d = 1$$

Соответствующие ортогональные полиномы:
полиномы Чебышева второго рода

$$T_m^*(x) = \frac{\sin((M+1)\arccos(x))}{\sin(\arccos(x))}$$

Узлы:

$$z_m = \cos\left(\frac{\pi(m+1)}{M+1}\right), \quad m = 1, \dots, M$$

Веса:

$$w_m = \frac{\pi}{M+1} \sin^2\left(\frac{\pi(m+1)}{M+1}\right), \quad m = 1, \dots, M$$

M	C(M)
1	2.0e-01
2	4.1e-03
3	3.4e-05
4	1.5e-07
5	4.2e-10

$$7) \rho(x) = \sqrt{\frac{x}{1-x}}, \quad c = 0, \quad d = 1$$

Соответствующие ортогональные полиномы:

$$\frac{1}{\sqrt{x}} T_{2M+1}(\sqrt{x}).$$

Узлы:

$$z_m = \cos^2\left(\frac{\pi(2m-1)}{2(2M+1)}\right), \quad m = 1, \dots, M$$

Веса:

$$w_m = \frac{2\pi}{2M+1} z_m, \quad m = 1, \dots, M$$

M	C(M)
1	4.9e-02
2	2.6e-04
3	5.3e-07
4	5.9e-10
5	4.1e-13

$$8) \rho(x) = -\ln(x), \quad c = 0, \quad d = 1$$

Соответствующие ортогональные полиномы:
полиномы ортогональные с весовой функцией $-\ln(x)$.

Таблица А9.

М	Узлы z_m , $m = 1, \dots, M$	Веса w_m , $m = 1, \dots, M$	C(M)
2	0.112009 0.602277	0.718539 0.281461	1.2e-04
3	0.063891 0.368997 0.766880	0.513405 0.391980 0.094615	2.4e-07
4	0.041448 0.245275 0.556165 0.848982	0.383464 0.386875 0.190435 0.039225	2.5e-10

9) $\rho(x) = \exp(-x)$, $c = 0$, $d = \infty$

Соответствующие ортогональные полиномы:
полиномы Лагерра $L_M(x)$.

Таблица A10.

M	Узлы z_m , $m = 1, \dots, M$	Веса w_m , $m = 1, \dots, M$	$w_m \exp(z_m)$, $m = 1, \dots, M$	C(M)
2	0.585786 3.414214	8.535533e-01 1.464466e-01	1.533326 4.450957	1.6e-01
3	0.415775 2.294280 6.289945	7.110930e-01 2.785177e-01 1.038926e-02	1.077693 2.762143 5.601095	5.0e-02
4	0.322548 1.745761 4.536620 9.395071	6.031541e-01 3.574187e-01 3.888790e-02 5.392947e-04	0.832739 2.048102 3.631146 6.487145	1.4e-02
5	0.263560 1.413403 3.596426 7.085810 12.640801	5.217556e-01 3.986668e-01 7.594245e-02 3.611759e-03 2.336997e-05	0.679094 1.638488 2.769443 4.315657 7.219186	4.0e-03
6	0.222847 1.188932 2.992736 5.775144 9.837467 15.982874	4.589647e-01 4.170008e-01 1.133734e-01 1.039920e-02 2.610172e-04 8.985479e-07	0.573536 1.369253 2.260685 3.350525 4.886827 7.849016	1.1e-03

Таблица А10 (продолжение).

M	Узлы z_m , $m = 1, \dots, M$	Веса w_m , $m = 1, \dots, M$	$w_m \exp(z_m)$, $m = 1, \dots, M$	C(M)
7	0.193044	4.093190e-01	0.496478	2.9e-04
	1.026665	4.218313e-01	1.177643	
	2.567877	1.471263e-01	1.918250	
	4.900353	2.063351e-02	2.771849	
	8.182153	1.074010e-03	3.841249	
	12.734180	1.586546e-05	5.380678	
	19.395728	3.170315e-08	8.405432	
8	0.170280	3.691886e-01	0.437723	7.7e-05
	0.903702	4.187868e-01	1.033869	
	2.251087	1.757950e-01	1.669710	
	4.266700	3.334349e-02	2.376925	
	7.045905	2.794536e-03	3.208541	
	10.758516	9.076509e-05	4.268576	
	15.740679	8.485747e-07	5.818083	
	22.863132	1.048001e-09	8.906226	
9	0.152322	3.361264e-01	0.391431	2.1e-05
	0.807220	4.112140e-01	0.921805	
	2.005135	1.992875e-01	1.480128	
	3.783474	4.746056e-02	2.086771	
	6.204957	5.599627e-03	2.772921	
	9.372985	3.052498e-04	3.591626	
	13.466237	6.592123e-06	4.648766	
	18.833598	4.110769e-08	6.212275	
	26.374072	3.290874e-11	9.363218	
10	0.137793	3.084411e-01	0.354010	5.4e-06
	0.729455	4.011110e-01	0.831902	
	1.808343	2.180683e-01	1.330289	
	3.401433	6.208746e-02	1.863064	
	5.552496	9.501517e-03	2.450256	
	8.330153	7.530084e-04	3.122764	
	11.843786	2.825923e-05	3.934153	
	16.279258	4.249314e-07	4.992415	
	21.996586	1.839565e-09	6.572202	
	29.920697	9.911827e-13	9.784696	

$$10) \rho(x) = \exp(-x^2), \quad c = -\infty, \quad d = \infty$$

Соответствующие ортогональные полиномы:
полиномы Эрмита $H_M(x)$.

Таблица A11.

M	Узлы z_m , $m = 1, \dots, M$	Веса w_m , $m = 1, \dots, M$	$w_m \exp(z_m^2)$, $m = 1, \dots, M$	C(M)
2	-0.707107 0.707107	8.862269e-01 8.862269e-01	1.461141 1.461141	3.7e-02
3	-1.224745 0.000000 1.224745	2.954090e-01 1.181636e+00 2.954090e-01	1.323931 1.181636 1.323931	1.8e-03
4	-1.650680 -0.524648 0.524648 1.650680	8.131284e-02 8.049141e-01 8.049141e-01 8.131284e-02	1.240226 1.059964 1.059964 1.240226	6.6e-05
5	-2.020183 -0.958572 0.000000 0.958572 2.020183	1.995324e-02 3.936193e-01 9.453087e-01 3.936193e-01 1.995324e-02	1.181489 0.986581 0.945309 0.986581 1.181489	1.8e-06
6	-2.350605 -1.335849 -0.436077 0.436077 1.335849 2.350605	4.530010e-03 1.570673e-01 7.246296e-01 7.246296e-01 1.570673e-01 4.530010e-03	1.136908 0.935581 0.876401 0.876401 0.935581 1.136908	4.2e-08

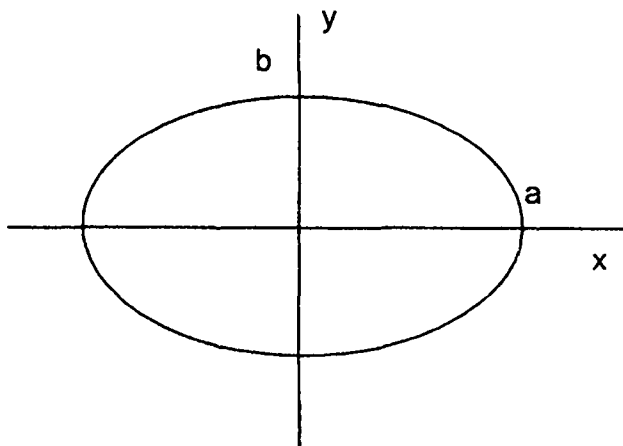
ПРИЛОЖЕНИЕ В

Преобразование некоторых регулярных областей интегрирования в прямоугольные области

1. Круг $S: x^2 + y^2 \leq R^2$

$$\iint_S f(x, y) dx dy = \int_0^{2\pi} \int_0^R r f(r \cos \varphi, r \sin \varphi) dr d\varphi$$

2. Эллипс $S: \frac{x^2}{a^2} + \frac{y^2}{b^2} \leq 1$



$$\iint_S f(x, y) dx dy = ab \int_0^{2\pi} \int_0^1 r f(ar \cos \varphi, br \sin \varphi) dr d\varphi$$

3. Сфера $S : x^2 + y^2 + z^2 \leq R^2$

$$\iiint_S f(x, y, z) dx dy dz = \int_0^\pi \int_0^{2\pi} \int_0^R r^2 \sin \theta \cdot f \begin{pmatrix} r \sin \theta \cdot \cos \varphi, \\ r \sin \theta \cdot \sin \varphi, \\ r \cos \theta \end{pmatrix} dr d\varphi d\theta$$

4. Эллипсоид $S : \frac{x^2}{a^2} + \frac{y^2}{b^2} + \frac{z^2}{c^2} \leq 1$

$$\iiint_S f(x, y, z) dx dy dz = abc \int_0^\pi \int_0^{2\pi} \int_0^1 r^2 \sin \theta \cdot f \begin{pmatrix} ar \sin \theta \cdot \cos \varphi, \\ br \sin \theta \cdot \sin \varphi, \\ cr \cos \theta \end{pmatrix} dr d\varphi d\theta$$

5. Круговой цилиндр $S : x^2 + y^2 \leq R^2, 0 \leq z \leq d$

$$\iiint_S f(x, y, z) dx dy dz = \int_0^d \int_0^{2\pi} \int_0^R r f(r \cos \varphi, r \sin \varphi, z) dr d\varphi dz$$

6. Эллиптический цилиндр $S : \frac{x^2}{a^2} + \frac{y^2}{b^2} \leq 1, 0 \leq z \leq d$

$$\iiint_S f(x, y, z) dx dy dz = ab \int_0^d \int_0^{2\pi} \int_0^1 r f(ar \cos \varphi, br \sin \varphi, z) dr d\varphi dz$$

ПРИЛОЖЕНИЕ С

Условия устойчивости для некоторых разностных схем Рунге-Кутты, Адамса и предиктор-корректор

Это приложение содержит информацию об областях устойчивости различных разностных схем для модельного уравнения

$$\frac{du}{dx} = -\gamma u, \gamma > 0, x \geq 0$$

$$u(0) = \alpha$$

Условие устойчивости имеет вид $\gamma h \leq \beta$, и следующая таблица представляет значения параметра β . Знак ∞ означает безусловную устойчивость.

Порядок аппроксимации, p	1	2	3	4
p-стадийная явная схема Рунге-Кутты	2	2	≈ 2.512	≈ 2.785
Явная схема Адамса	2	1	≈ 0.54	≈ 0.3
Неявная схема Адамса	∞	∞	≈ 6	≈ 3
Схема предиктор - корректор		2	≈ 1.72	≈ 1.41

ПРИЛОЖЕНИЕ D

Краткий обзор программного обеспечения

В этом приложении приводится некоторая информация о библиотеках программ, широко используемых на практике. Здесь приводится только краткое описание вычислительных процедур, имеющих отношение к темам рассмотренным в этой книге. Более подробную информацию можно найти в соответствующей документации.

Библиотека подпрограмм LAPACK для задач линейной алгебры

LAPACK (Linear Algebra PACKage) представляет собой библиотеку подпрограмм для решения наиболее часто встречающихся задач линейной алгебры. Имеются две версии этой библиотеки: на языках Fortran 77(90) и C. Библиотека LAPACK не является коммерческим продуктом и может быть загружена из архива Netlib (www.netlib.org/lapack). Она включает в себя подпрограммы для

- решения систем линейных уравнений
- нахождения решения переопределённых систем линейных уравнений в смысле наименьших квадратов
- решения задач на собственное значение

Подпрограммы в библиотеке LAPACK подразделяются на три типа:

Подпрограммы верхнего уровня

- решают некоторую задачу, как, например, нахождение решения системы линейных уравнений или собственных значений матрицы

Подпрограммы верхнего уровня с дополнительными возможностями

- обеспечивают больше возможностей/информации по сравнению с подпрограммами верхнего уровня. Например, они позволяют

оценить точность полученного решения или сбалансировать матрицу для повышения точности

Подпрограммы нижнего уровня

- вызываются подпрограммами верхнего уровня и выполняют определённые вычислительные процедуры, как, например, LU разложение или приведение вещественной симметричной матрицы к трёхдиагональной матрице

Ниже приведены основные имена подпрограмм. Реальные имена образуются следующим образом: (приставка)(основное имя), где приставка принимает одно из следующих значений:

s - вещественные значения, одинарная точность

d - вещественные значения, двойная точность

c - комплексные значения, одинарная точность

z - комплексные значения, двойная точность

Подпрограммы верхнего уровня для решения систем линейных уравнений

gesv - решение систем линейных уравнений с матрицей общего вида

gbsv - решение систем линейных уравнений с ленточной матрицей общего вида

gtsv - решение систем линейных уравнений с трёхдиагональной матрицей общего вида

posv - решение систем линейных уравнений с симметричной/Эрмитовой положительно определённой матрицей

ppsv - решение систем линейных уравнений с симметричной/Эрмитовой положительно определённой матрицей (хранение в упакованном виде)

pbsv - решение систем линейных уравнений с симметричной/Эрмитовой положительно определённой ленточной матрицей

ptsv - решение систем линейных уравнений с симметричной/Эрмитовой положительно определённой трёхдиагональной матрицей

sysv/hesv - решение систем линейных уравнений с симметричной/Эрмитовой/ комплексной неопределённой матрицей

spsv/hpsv - решение систем линейных уравнений с симметричной/Эрмитовой/ комплексной неопределённой матрицей (хранение в упакованном виде)

Подпрограммы верхнего уровня с дополнительными возможностями для решения систем линейных уравнений

gesvx - решение систем линейных уравнений с матрицей общего вида. Обеспечиваются оценки числа обусловленности и ошибки полученного решения

gbsvx - решение систем линейных уравнений с ленточной матрицей общего вида. Обеспечиваются оценки числа обусловленности и ошибки полученного решения

gtsvx - решение систем линейных уравнений с трёхдиагональной матрицей общего вида. Обеспечиваются оценки числа обусловленности и ошибки полученного решения

posvx - решение систем линейных уравнений с симметричной/Эрмитовой положительно определённой матрицей. Обеспечиваются оценки числа обусловленности и ошибки полученного решения

ppsvx - решение систем линейных уравнений с симметричной/Эрмитовой положительно определённой матрицей (хранение в упакованном виде). Обеспечиваются оценки числа обусловленности и ошибки полученного решения

pbsvx - решение систем линейных уравнений с симметричной/Эрмитовой положительно определённой ленточной матрицей. Обеспечиваются оценки числа обусловленности и ошибки полученного решения

ptsvx - решение систем линейных уравнений с симметричной/Эрмитовой положительно определённой трёхдиагональной матрицей. Обеспечиваются оценки числа обусловленности и ошибки полученного решения

sysvx/hesvx - решение систем линейных уравнений с симметричной/Эрмитовой/комплексной неопределённой матрицей. Обеспечиваются оценки числа обусловленности и ошибки полученного решения

spsvx/hpsvx - решение систем линейных уравнений с симметричной/Эрмитовой/комплексной неопределённой матрицей (хранение в упакованном виде). Обеспечиваются оценки числа обусловленности и ошибки полученного решения

Подпрограммы верхнего уровня для стандартной задачи на собственное значение

syev/heev - вычисление всех собственных значений и (если требуется) собственных векторов вещественной симметричной/Эрмитовой матрицы

spev/hpev - вычисление всех собственных значений и (если требуется) собственных векторов вещественной симметричной/Эрмитовой матрицы (хранение в упакованном виде)

sbev/hbev - вычисление всех собственных значений и (если требуется) собственных векторов вещественной симметричной/Эрмитовой ленточной матрицы

stev - вычисление всех собственных значений и (если требуется) собственных векторов вещественной симметричной трёхдиагональной матрицы

gees - вычисление собственных значений, формы Шура и (если требуется) матрицы векторов Шура несимметричной матрицы

geev - вычисление собственных значений и (если требуется) левых и/или правых собственных векторов несимметричной матрицы

syevd/heevd - вычисление всех собственных значений и (если требуется) собственных векторов вещественной симметричной/Эрмитовой матрицы. Для вычисления собственных векторов используется алгоритм «разделяй и властвуй»

spevd/hpevd - вычисление всех собственных значений и (если требуется) собственных векторов вещественной симметричной/Эрмитовой матрицы (хранение в упакованном виде). Для вычисления собственных векторов используется алгоритм «разделяй и властвуй».

sbevd/hbevd - вычисление всех собственных значений и (если требуется) собственных векторов вещественной симметричной/Эрмитовой ленточной матрицы. Для вычисления собственных векторов используется алгоритм «разделяй и властвуй».

stevd - вычисление всех собственных значений и (если требуется) собственных векторов вещественной симметричной трёхдиагональной матрицы. Для вычисления собственных векторов используется алгоритм «разделяй и властвуй»

Подпрограммы верхнего уровня с дополнительными возможностями для стандартной задачи на собственное значение

syevx/heevx - вычисление всех собственных значений и (если требуется) собственных векторов вещественной симметричной/Эрмитовой матрицы. Требуемые собственные значения и вектора могут быть выбраны по заданному интервалу величин или интервалу номеров

spevx/hpevx - вычисление всех собственных значений и (если требуется) собственных векторов вещественной симметричной/Эрмитовой матрицы (хранение в упакованном виде). Требуемые собственные значения и вектора могут быть выбраны по заданному интервалу величин или интервалу номеров

sbevx/hbevx - вычисление всех собственных значений и (если требуется) собственных векторов вещественной симметричной/Эрмитовой ленточной матрицы. Требуемые собственные значения и вектора могут быть выбраны по заданному интервалу величин или интервалу номеров

stevx - вычисление всех собственных значений и (если требуется) собственных векторов вещественной симметричной трёхдиагональной матрицы. Требуемые собственные значения и вектора могут быть выбраны по заданному интервалу величин или интервалу номеров

geesx - вычисление собственных значений, формы Шура и (если требуется) матрицы векторов Шура несимметричной матрицы. Дополнительно, можно расположить собственные значения на диагонали формы Шура таким образом, что требуемые собственные значения расположены в левом верхнем углу

geevx - вычисление собственных значений и (если требуется) левых и/или правых собственных векторов несимметричной матрицы. Дополнительно, можно осуществить балансировку матрицы для улучшения обусловленности и вычислить обратное число обусловленности для собственных значений и векторов

Подпрограммы верхнего уровня для обобщённой задачи на собственное значение

sygv/hegv - вычисление всех собственных значений и (если требуется) собственных векторов для обобщённой проблемы на собственное

значение вида $\mathbf{Ax}=\lambda\mathbf{Mx}$, где $\mathbf{A}$ и $\mathbf{M}$ вещественные симметричные/Эрмитовы матрицы и матрица $\mathbf{M}$ также положительно определённая

spgv/hpgv - вычисление всех собственных значений и (если требуется) собственных векторов для обобщённой проблемы на собственное значение вида $\mathbf{Ax}=\lambda\mathbf{Mx}$, где $\mathbf{A}$ и $\mathbf{M}$ вещественные симметричные/Эрмитовы матрицы и матрица $\mathbf{M}$ также положительно определённая (хранение в упакованном виде)

sbgv/hbgv - вычисление всех собственных значений и (если требуется) собственных векторов для обобщённой проблемы на собственное значение вида $\mathbf{Ax}=\lambda\mathbf{Mx}$, где $\mathbf{A}$ и $\mathbf{M}$ ленточные вещественные симметричные/Эрмитовы матрицы и матрица $\mathbf{M}$ также положительно определённая

Библиотека подпрограмм IMSL®

Библиотека IMSL (International Mathematics and Statistics Libraries) широко используется на практике и включает в себя библиотеки численных алгоритмов для языков C, Fortran 90, Fortran 77 и Java.

Библиотека IMSL C (CNL)

Библиотека CNL написана на языке C на основе подпрограмм из библиотеки IMSL Fortran. Весия 4.0 этой библиотеки содержит более 300 математических и статистических алгоритмов.

Библиотека IMSL Fortran 90 MP (F90 MP)

Эта библиотека основана на алгоритмах оптимизированных для использования в многопроцессорных системах. Библиотека F90 MP включает в себя:

- функциональные модули для объектно-ориентированного программирования
- подпрограммы, написанные на языке Fortran 90 для более эффективного выполнения программ
- библиотеку IMSL Fortran 77

Библиотека IMSL for Java (JNL)

Библиотека JNL включает в себя функции отсутствующие в языке Java: класс численного типа, комплексный класс и три класса численных функций (класс специальных функций, класс линейной алгебры и класс статистики).

Библиотека IMSL Fortran 77 (FNL)

Библиотека FNL состоит из более чем 900 подпрограмм написанных на языке Fortran 77, предназначенных для решения широкого класса задач прикладной математики и статистического анализа. Все ведущие производители компиляторов языка Fortran включают эту библиотеку в состав своих программных продуктов. Далее приводится краткое

описание подпрограмм библиотеки FNL, имеющих отношение к темам рассмотренным в этой книге.

Решение систем линейных уравнений

lsacb - решение систем линейных уравнений с комплексной ленточной матрицей (применяется итерационное уточнение решения)

lsacg - решение систем линейных уравнений с комплексной матрицей общего вида (применяется итерационное уточнение решения)

lsadh - решение систем линейных уравнений с Эрмитовой положительно определённой матрицей (применяется итерационное уточнение решения)

lsads - решение систем линейных уравнений с вещественной симметричной положительно определённой матрицей (применяется итерационное уточнение решения)

lsahf - решение систем линейных уравнений с Эрмитовой матрицей (применяется итерационное уточнение решения)

lsaqh - решение систем линейных уравнений с Эрмитовой ленточной положительно определённой матрицей

lsaqs - решение систем линейных уравнений с ленточной вещественной симметричной положительно определённой матрицей

lsarb - решение систем линейных уравнений с ленточной вещественной матрицей (применяется итерационное уточнение решения)

lsarg - решение систем линейных уравнений с вещественной матрицей общего вида (применяется итерационное уточнение решения)

lsasf - решение систем линейных уравнений с вещественной симметричной матрицей (применяется итерационное уточнение решения)

lsacb - решение систем линейных уравнений с комплексной ленточной матрицей

lsclg - решение систем линейных уравнений с комплексной матрицей общего вида

lsclq - вычисление LDU разложения комплексной трёхдиагональной матрицы

lsclr - вычисление LDU разложения вещественной трёхдиагональной матрицы

lsclt - решение систем линейных уравнений с комплексной треугольной матрицей

lsldh - решение систем линейных уравнений с Эрмитовой положительно определённой матрицей

lslds - решение систем линейных уравнений с вещественной симметричной положительно определённой матрицей

lsllhf - решение систем линейных уравнений с Эрмитовой матрицей

lsllpb - вычисление разложения Холецкого вещественной симметричной положительно определённой матрицы

lsllqb - вычисление разложения Холецкого Эрмитовой положительно определённой матрицы

lsllqh - решение систем линейных уравнений с Эрмитовой ленточной положительно определённой матрицей

lsllqs - решение систем линейных уравнений с вещественной симметричной ленточной положительно определённой матрицей

lsllrb - решение систем линейных уравнений с вещественной ленточной матрицей

lsllrg - решение систем линейных уравнений с вещественной матрицей общего вида

lsllrt - решение систем линейных уравнений с вещественной треугольной матрицей

lsllsf - решение систем линейных уравнений с вещественной симметричной матрицей

lslltc - решение систем линейных уравнений с комплексной Тёплицевой матрицей

lsllto - решение систем линейных уравнений с вещественной Тёплицевой матрицей

lslltq - решение систем линейных уравнений с комплексной трёхдиагональной матрицей

lslltr - решение систем линейных уравнений с вещественной трёхдиагональной матрицей

lsllxd - решение систем линейных уравнений с разреженной вещественной симметричной положительно определённой матрицей

lsllxg - решение систем линейных уравнений с разреженной вещественной матрицей методом исключения Гаусса

lsllzd - решение систем линейных уравнений с разреженной Эрмитовой положительно определённой матрицей методом исключения Гаусса

lsllzg - решение систем линейных уравнений с разреженной комплексной матрицей методом исключения Гаусса

Задача на собственное значение

evahf - вычисление наибольшего или наименьшего по модулю собственного значения Эрмитовой матрицы

evasb - вычисление наибольшего или наименьшего по модулю собственного значения вещественной ленточной симметричной матрицы

evasf - вычисление наибольшего или наименьшего по модулю собственного значения вещественной симметричной матрицы

evbhf - вычисление собственных значений Эрмитовой матрицы, лежащих в заданном интервале

evbsb - вычисление собственных значений вещественной ленточной симметричной матрицы, лежащих в заданном интервале

evbsf - вычисление собственных значений вещественной симметричной матрицы, лежащих в заданном интервале

evcsg - вычисление всех собственных значений и векторов комплексной матрицы

evcch - вычисление всех собственных значений и векторов комплексной верхней матрицы Хессенберга

evchf - вычисление всех собственных значений и векторов комплексной Эрмитовой матрицы

everg - вычисление всех собственных значений и векторов вещественной матрицы

everh - вычисление всех собственных значений и векторов вещественной верхней матрицы Хессенберга

evcsb - вычисление всех собственных значений и векторов вещественной ленточной симметричной матрицы

evcsf - вычисление всех собственных значений и векторов вещественной симметричной матрицы

evehf - вычисление наибольшего или наименьшего по модулю собственного значения и соответствующих собственных векторов Эрмитовой матрицы

evesb - вычисление наибольшего или наименьшего по модулю собственного значения и соответствующих собственных векторов вещественной ленточной симметричной матрицы

evesf - вычисление наибольшего или наименьшего по модулю собственного значения и соответствующих собственных векторов вещественной симметричной матрицы

evfhf - вычисление собственных значений Эрмитовой матрицы, лежащих в заданном интервале и соответствующих собственным векторам

evfsb - вычисление собственных значений вещественной ленточной симметричной матрицы, лежащих в заданном интервале и соответствующих собственным векторам

evfsf - вычисление собственных значений вещественной симметричной матрицы, лежащих в заданном интервале и соответствующих собственным векторам

evlrg - вычисление всех собственных значений комплексной матрицы

evlch - вычисление всех собственных значений комплексной верхней матрицы Хессенберга

evlhf - вычисление всех собственных значений комплексной Эрмитовой матрицы

evlrg - вычисление всех собственных значений вещественной матрицы

evlrh - вычисление всех собственных значений вещественной верхней матрицы Хессенберга

evlsb - вычисление всех собственных значений вещественной ленточной симметричной матрицы

evlsf - вычисление всех собственных значений вещественной симметричной матрицы

gvccg - вычисление всех собственных значений и собственных векторов для обобщённой проблемы на собственное значение вида $Ax = \lambda Mx$, где **A** и **M** комплексные матрицы

gvcrq - вычисление всех собственных значений и собственных векторов для обобщённой проблемы на собственное значение вида $Ax = \lambda Mx$, где **A** и **M** вещественные матрицы

gvcsf - вычисление всех собственных значений и собственных векторов для обобщённой проблемы на собственное значение вида $Ax = \lambda Mx$, где **A** и **M** вещественные симметричные матрицы и матрица **M** также положительно определённая

gvleg - вычисление всех собственных значений для обобщённой проблемы на собственное значение вида $Ax = \lambda Mx$, где **A** и **M** комплексные матрицы

gvlrg - вычисление всех собственных значений для обобщённой проблемы на собственное значение вида $Ax = \lambda Mx$, где **A** и **M** вещественные матрицы

gvlsf - вычисление всех собственных значений для обобщённой проблемы на собственное значение вида $Ax = \lambda Mx$, где **A** и **M**

вещественные симметричные матрицы и матрица **M** также положительно определённая

Решение нелинейных уравнений

zanly - вычисление нуля комплексной аналитической функции

zbren - вычисление нуля функции, которая меняет знак на заданном интервале

zplrc - вычисление нулей полинома с вещественными коэффициентами (метод Лагерра)

zprocc - вычисление нулей полинома с комплексными коэффициентами

zporc - вычисление нулей полинома с вещественными коэффициентами (метод Дженкинса-Трауба)

zreal - вычисление нуля вещественной функции

neqbf - решение систем нелинейных уравнений (метод Бroyдена, начальная матрица Якоби задаётся приближённо)

neqbj - решение систем нелинейных уравнений (метод Бroyдена, начальная матрица Якоби задаётся точно)

neqnf - решение систем нелинейных уравнений с использованием алгоритма Пауэлла (модификация метода Ньютона), матрица Якоби задаётся приближённо

neqnj - решение систем нелинейных уравнений с использованием алгоритма Пауэлла (модификация метода Ньютона), матрица Якоби задаётся точно

Численное интегрирование

qand - вычисление интеграла от N-мерной функции по гиперпараллелепипеду ($N \leq 20$)

twodq - вычисление двумерного интеграла

qdag - вычисление интеграла методом Гаусса-Кронрода

qdagl - вычисление интеграла по бесконечному или полубесконечному интервалу

qdagp - вычисление интеграла от функции, имеющей особенности в заданных точках отрезка интегрирования

qdagss - вычисление интеграла от функции, имеющей особенности на концах отрезка интегрирования

qdawc - вычисление главного значения интеграла в смысле Коши от функции $f(x)/(x - c)$

qdawf - вычисление интеграла Фурье

qdawo - вычисление интеграла от функции содержащей $\sin(x)$ или $\cos(x)$

qdaws - вычисление интеграла от функции имеющей особенность вида $(x-a)^{\alpha} \ln(x-a)$ или $(b-x)^{\beta} \ln(b-x)$, где a и b пределы интегрирования

qdnng - вычисление интеграла от гладкой функции

gorcf - вычисление узлов и весов формул Гаусса, Гаусса-Радау или Гаусса-Лобатто, когда соответствующие ортогональные полиномы заданы в форме трёхчленных рекуррентных соотношений

gorul - вычисление узлов и весов формул Гаусса, Гаусса-Радау или Гаусса-Лобатто для некоторых классических весовых функций

Решение обыкновенных дифференциальных уравнений

ivpag - решение задачи Коши (неявные схемы Адамса или схемы BDF)

ivprk - решение задачи Коши (схемы Рунге-Кутта пятого и шестого порядка)

ivmrk - решение задачи Коши (встроенные Рунге-Кутта пары различных порядков)

bvpfd - решение системы дифференциальных уравнений с граничными условиями в двух точках методом конечных разностей

bvpms - решение системы дифференциальных уравнений с граничными условиями в двух точках методом стрельбы

Интерполяция и аппроксимация функций

csakm - построение кубического сплайна Акима

csdec - построение кубического сплайна с заданными условиями на концах отрезка

csder - вычисление производных кубического сплайна в заданных точках

csher - построение кубического сплайна Эрмита

csiez - построение кубического сплайна

csint - построение кубического сплайна, условия на концах отрезка определяются программой

csitg - вычисление интеграла от кубического сплайна

csper - построение кубического сплайна с периодическими условиями на концах отрезка

surf - построение двумерного интерполяционного полинома
csval - вычисление значения кубического сплайна в заданной точке
fnlsq - приближение набора данных заданной функцией (метод наименьших квадратов)
rcurv - приближение набора данных полиномом (метод наименьших квадратов)
rline - линейное приближение набора данных (метод наименьших квадратов)

Библиотеки подпрограмм Группы по численным алгоритмам (NAG[®], The Numerical Algorithms Group)

Библиотека NAG f90

Библиотека NAG f90 состоит из более чем 200 основных подпрограмм, реализующих различные алгоритмы. Все подпрограммы написаны на языке Fortran 90/95, что даёт простой и эффективный инструмент для научных и инженерных расчётов.

Решение систем линейных уравнений

nag_gen_lin_sol - решение систем линейных уравнений с вещественной или комплексной матрицей общего вида
nag_gen_lin_fac - вычисление LU разложения вещественной или комплексной матрицы общего вида
nag_gen_lin_sol_fac - решение систем линейных уравнений с вещественной или комплексной матрицей общего вида, которая предварительно была разложена с использованием процедуры **nag_gen_lin_fac**
nag_sym_lin_sol - решение систем линейных уравнений с вещественной симметричной или комплексной Эрмитовой матрицей
nag_sym_lin_fac - вычисление разложения Холецкого или Банча-Кауфмана (Bunch –Kaufman) вещественной симметричной или комплексной Эрмитовой матрицы
nag_sym_lin_sol_fac - решение систем линейных уравнений с вещественной симметричной или комплексной Эрмитовой матрицей,

которая предварительно была разложена с использованием процедуры

nag_sym_lin_fac

nag_tri_lin_sol - решение систем линейных уравнений с вещественной или комплексной треугольной матрицей

nag_tri_lin_cond - оценка числа обусловленности вещественной или комплексной треугольной матрицы

nag_tri_mat_det - вычисление определителя вещественной или комплексной треугольной матрицы

nag_gen_bnd_lin_sol - решение систем линейных уравнений с вещественной или комплексной ленточной матрицей

nag_gen_bnd_lin_fac - вычисление LU разложения вещественной или комплексной ленточной матрицы

nag_gen_bnd_lin_sol_fac - решение систем линейных уравнений с вещественной или комплексной ленточной матрицей, которая предварительно была разложена с использованием процедуры **nag_gen_bnd_lin_fac**

nag_sym_bnd_lin_sol - решение систем линейных уравнений с вещественной симметричной или комплексной Эрмитовой положительно определённой ленточной матрицей

nag_sym_bnd_lin_fac - вычисление разложения Холецкого вещественной симметричной или комплексной Эрмитовой положительно определённой ленточной матрицы

nag_sym_bnd_lin_sol_fac - решение систем линейных уравнений с вещественной симметричной или комплексной Эрмитовой положительно определённой ленточной матрицей, которая предварительно была разложена с использованием процедуры **nag_sym_bnd_lin_fac**

Задача на собственное значение

nag_sym_eig_all - вычисление всех собственных значений и (если требуется) собственных векторов вещественной симметричной или комплексной Эрмитовой матрицы

nag_sym_eig_sel - вычисление отдельных собственных значений и (если требуется) соответствующих собственных векторов вещественной симметричной или комплексной Эрмитовой матрицы

nag_sym_tridiag_reduc - преобразование вещественной симметричной или комплексной Эрмитовой матрицы в вещественную симметричную трёхдиагональную матрицу

nag_sym_tridiag_orth – построение матрицы преобразования, заданного процедурой **nag_sym_tridiag_reduc**

nag_sym_tridiag_eig_all - вычисление всех собственных значений и (если требуется) собственных векторов вещественной симметричной трёхдиагональной матрицы

nag_sym_tridiag_eig_val - вычисление отдельных собственных значений вещественной симметричной трёхдиагональной матрицы

nag_sym_tridiag_eig_vec - вычисление отдельных собственных векторов вещественной симметричной трёхдиагональной матрицы

nag_nsym_eig_all - вычисление всех собственных значений и (если требуется) собственных векторов вещественной или комплексной матрицы общего вида

nag_schur_fac - вычисление разложения Шура вещественной или комплексной матрицы общего вида

nag_sym_gen_eig_all - вычисление всех собственных значений и (если требуется) собственных векторов для обобщённой проблемы на собственное значение вида $Ax = \lambda Mx$, где **A** и **M** вещественные симметричные или комплексные Эрмитовы матрицы и матрица **M** также положительно определённая

nag_sym_gen_eig_sel - вычисление отдельных собственных значений и (если требуется) соответствующих собственных векторов для обобщённой проблемы на собственное значение вида $Ax = \lambda Mx$, где **A** и **M** вещественные симметричные или комплексные Эрмитовы матрицы и матрица **M** также положительно определённая

nag_nsym_gen_eig_all - вычисление всех собственных значений и (если требуется) собственных векторов для обобщённой проблемы на собственное значение вида $Ax = \lambda Mx$, где **A** и **M** вещественные или комплексные несимметричные матрицы

nag_gen_schur_fac - вычисление обобщённого разложения Шура вещественной или комплексной матрицы

Решение нелинейных уравнений

nag_polynom_roots - вычисление нулей полинома

nag_nlin_eqn_sol - решение нелинейного уравнения

nag_nlin_sys_sol - решение систем нелинейных уравнений

Численное интегрирование

nag_quad_1d_gen - вычисление интеграла от функции, имеющей особенности в заданных точках отрезка интегрирования

nag_quad_1d_wt_trig - вычисление интеграла от функции содержащей $\sin(x)$ или $\cos(x)$

nag_quad_1d_wt_end_sing - вычисление интеграла от функции имеющей особенность вида $(x-a)^{\alpha}\ln(x-a)$ или $(b-x)^{\beta}\ln(b-x)$, где a и b пределы интегрирования

nag_quad_1d_wt_hilb - вычисление главного значения интеграла в смысле Коши от функции $f(x)/(x - c)$

nag_quad_1d_data - вычисление интеграла от функции, определённой набором значений, метод Гилла-Миллера (Gill-Miller)

nag_quad_1d_inf_gen - вычисление интеграла по бесконечному или полубесконечному интервалу

nag_quad_1d_inf_wt_trig - вычисление интеграла по или полубесконечному интервалу от функции содержащей $\sin(x)$ или $\cos(x)$

nag_quad_md_rect_qand - вычисление интеграла от N-мерной функции по гиперпараллелепипеду

nag_quad_2d - twodq - вычисление двумерного интеграла

nag_quad_gs_wt_absc - gorul - вычисление узлов и весов формул Гаусса-Кристоффеля для некоторых весовых функций

Решение обыкновенных дифференциальных уравнений (методы Рунге-Кутты)

nag_rk_interval - решение задачи Коши на заданном интервале

nag_rk_global_err - оценка глобальной ошибки приближённого решения

nag_rk_step - вычисление одного шага разностной схемы

nag_rk_interp - интерполяция приближённого решения

Интерполяция и аппроксимация функций

nag_pch_monot_interp - построение кусочно-кубического интерполяционного полинома Эрмита

nag_pch_eval - вычисление значения и (если требуется) производной кусочно-кубического интерполяционного полинома Эрмита

nag_pch_intg - вычисление интеграла от кусочно-кубического интерполяционного полинома Эрмита

nag_spline_1d_auto_fit - построение кубической сплайн функции, аппроксимирующей произвольный одномерный набор данных

nag_spline_1d_lsq_fit - построение кубической сплайн функции, аппроксимирующей произвольный одномерный набор данных методом наименьших квадратов

nag_spline_1d_interp - построение интерполяционной кубической сплайн функции для произвольного одномерного набора данных

nag_spline_1d_eval - вычисление значения и (если требуется) первых трёх производных кубической сплайн функции

nag_spline_1d_intg - вычисление интеграла от кубической сплайн функции

nag_spline_2d_auto_fit - построение бикубической сплайн функции, аппроксимирующей произвольный двумерный набор данных

nag_spline_2d_lsq_fit - построение бикубической сплайн функции, аппроксимирующей произвольный двумерный набор данных методом наименьших квадратов

nag_spline_2d_interp - построение интерполяционной бикубической сплайн функции для двумерного набора данных, определённого в узлах прямоугольной сетки на плоскости

nag_spline_2d_eval - вычисление значения интерполяционной бикубической сплайн функции

nag_spline_2d_intg - вычисление интеграла от интерполяционной бикубической сплайн функции

nag_scat_2d_interp - построение двумерной интерполяционной функции модифицированным методом Шепарда (Shepard)

nag_scat_2d_eval - вычисление значения и частных производных двумерной интерполяционной функции

nag_scat_3d_interp - построение трёхмерной интерполяционной функции модифицированным методом Шепарда (Shepard)

nag_scat_3d_eval - вычисление значения и частных производных трёхмерной интерполяционной функции

Библиотека NAG C (Mark 5)

Библиотека The NAG C содержит около 400 процедур, предназначенных для решения различных вычислительных задач. Краткое

описание некоторых процедур приведено ниже. Вычислительные функции библиотеки NAG C имеют два имени: короткое, состоящее из шести знаков, и более длинное описательное, например, `f02awc` или `nag_hermitian_eigenvalues`. Все процедуры этой библиотеки используют вычисления с двойной точностью.

Решение систем линейных уравнений

f01bnc (`nag_complex_cholesky`) - вычисление UU^H разложения комплексной Эрмитовой положительно определённой матрицы

f01mcc (`nag_real_cholesky_skyline`) - вычисление LDL^T разложения вещественной симметричной положительно определённой ленточной матрицы с переменной шириной ленты

f03aec (`nag_real_cholesky`) - вычисление разложения Холецкого и определителя вещественной симметричной положительно определённой матрицы

f03afc (`nag_real_lu`) - вычисление LU разложения и определителя вещественной матрицы

f03ahc (`nag_complex_lu`) - вычисление LU разложения и определителя комплексной матрицы

f04adc (`nag_complex_lin_eqn_mult_rhs`) - решение систем линейных уравнений с комплексной матрицей общего вида (несколько векторов правой части)

f04agc (`nag_real_cholesky_solve_mult_rhs`) - решение систем линейных уравнений с вещественной симметричной положительно определённой матрицей, которая предварительно была разложена с использованием процедуры `nag_real_cholesky` (`f03aec`) (несколько векторов правой части)

f04ajc (`nag_real_lu_solve_mult_rhs`) - решение систем линейных уравнений с вещественной матрицей, которая предварительно была разложена с использованием процедуры `nag_real_lu` (`f03afc`) (несколько векторов правой части)

f04akc (`nag_complex_lu_solve_mult_rhs`) - решение систем линейных уравнений с комплексной матрицей, которая предварительно была разложена с использованием процедуры `nag_complex_lu` (`f03ahc`) (несколько векторов правой части)

f04arc (`nag_real_lin_eqn`) - решение систем линейных уравнений с вещественной матрицей (один вектор правой части)

f04awc (nag_hermitian_lin_eqn_mult_rhs) - решение систем линейных уравнений с комплексной Эрмитовой положительно определённой матрицей, которая предварительно была разложена с использованием процедуры **nag_complex_cholesky (f01bnc)** (несколько векторов правой части)

f04mcc (nag_real_cholesky_skyline_solve) - решение систем линейных уравнений с вещественной симметричной положительно определённой ленточной матрицей с переменной шириной ленты (матрица предварительно была разложена с использованием процедуры **nag_real_cholesky_skyline (f01mcc)**)

f11jac (nag_sparse_sym_chol_fac) - вычисление неполного разложения Холеского

f11jcc (nag_sparse_sym_chol_sol) - решение систем линейных уравнений с вещественной симметричной разреженной матрицей методом сопряжённых градиентов с предобуславливанием, основанном на неполном разложении Холеского

f11jec (nag_sparse_sym_sol) - решение систем линейных уравнений с вещественной симметричной разреженной матрицей методом сопряжённых градиентов с предобуславливанием Якоби, SSOR, или без предобуславливания

f1ldac (nag_sparse_nsym_fac) - вычисление неполного LU разложения

f1ldcc (nag_sparse_nsym_fac_sol) - решение систем линейных уравнений с вещественной несимметричной разреженной матрицей методом сопряжённых градиентов с предобуславливанием, основанном на неполном LU разложении

f1ldec (nag_sparse_nsym_sol) - решение систем линейных уравнений с вещественной несимметричной разреженной матрицей методом сопряжённых градиентов с предобуславливанием Якоби, SSOR, или без предобуславливания

Задача на собственное значение

f01qcc (nag_real_qr) - вычисление QR разложения вещественной матрицы размером M на N

f01rcc (nag_complex_qr) - вычисление QR разложения комплексной матрицы размером M на N

f02aac (nag_real_symm_eigenvalues) - вычисление всех собственных значений вещественной симметричной матрицы

f02abc (nag_real_symm_eigensystem) - вычисление всех собственных значений и собственных векторов вещественной симметричной матрицы

f02adc (nag_real_symm_general_eigenvalues) - вычисление всех собственных значений для обобщённой проблемы на собственное значение вида $Ax = \lambda Mx$, где **A** и **M** вещественные симметричные матрицы и матрица **M** также положительно определённая

f02aec (nag_real_symm_general_eigensystem) - вычисление всех собственных значений и собственных векторов для обобщённой проблемы на собственное значение вида $Ax = \lambda Mx$, где **A** и **M** вещественные симметричные матрицы и матрица **M** также положительно определённая

f02afc (nag_real_eigenvalues) - вычисление всех собственных значений вещественной матрицы

f02agc (nag_real_eigensystem) - вычисление всех собственных значений и собственных векторов вещественной матрицы

f02awc (nag_hermitian_eigenvalues) - вычисление всех собственных значений комплексной Эрмитовой матрицы

f02axc (nag_hermitian_eigensystem) - вычисление всех собственных значений и собственных векторов комплексной Эрмитовой матрицы

f02bjc (nag_real_general_eigensystem) - вычисление всех собственных значений и (если требуется) собственных векторов для обобщённой проблемы на собственное значение вида $Ax = \lambda Mx$, где **A** и **M** вещественные матрицы (QZ разложение)

f02ecc (nag_real_eigensystem_sel) - вычисление отдельных собственных значений и соответствующих собственных векторов вещественной матрицы

f02gcc (nag_complex_eigensystem_sel) - вычисление отдельных собственных значений и соответствующих собственных векторов комплексной матрицы

Решение нелинейных уравнений

c02afc (nag_zeros_complex_poly) - вычисление нулей полинома с комплексными коэффициентами

c02agc (nag_zeros_real_poly) - вычисление нулей полинома с вещественными коэффициентами

c05adc (nag_zero_cont_func_bd) - вычисление нуля вещественной непрерывной функции

c05nbc (nag_zero_nonlin_eqns) - решение систем нелинейных уравнений (метод Бroyдена, начальная матрица Якоби задаётся приближённо)

c05pbc (nag_zero_nonlin_eqns_deriv) - решение систем нелинейных уравнений (метод Бroyдена, начальная матрица Якоби задаётся точно)

Численное интегрирование

d01ajc (nag_1d_quad_gen) - вычисление одномерного интеграла

d01akc (nag_1d_quad_osc) - вычисление интеграла от осциллирующей функции

d01alc (nag_1d_quad_brkpts) - вычисление интеграла от функции, имеющей особенности в заданных точках отрезка интегрирования

d01amc (nag_1d_quad_inf) - вычисление интеграла по бесконечному или полубесконечному интервалу

d01anc (nag_1d_quad_wt_trig) - вычисление интеграла от функции содержащей $\sin(x)$ или $\cos(x)$

d01apc (nag_1d_quad_wt_alglog) - вычисление интеграла от функции имеющей особенность вида $(x-a)^{\alpha}\ln(x-a)$ или $(b-x)^{\beta}\ln(b-x)$, где a и b пределы интегрирования

d01aqc (nag_1d_quad_wt_cauchy) - вычисление главного значения интеграла в смысле Коши от функции $f(x)/(x-c)$

d01asc (nag_1d_quad_inf_wt_trig) - вычисление интеграла по полубесконечному интервалу от функции содержащей $\sin(x)$ или $\cos(x)$

d01bac (nag_1d_quad_gauss) - вычисление интеграла по формулам Гаусса-Кристоффеля (формулы Гаусса, Гаусса-Лагерра и Гаусса-Эрмита)

d01fcc (nag_multid_quad_adapt) - вычисление интеграла от N-мерной функции ($N \leq 15$)

d01gac (nag_1d_quad_vals) - вычисление интеграла от функции, определённой набором значений

d01gbc (nag_multid_quad_monte_carlo) - вычисление многомерного интеграла методом Монте Карло

Решение обыкновенных дифференциальных уравнений

d02cjc (nag_ode_ivp_adams_gen) - решение задачи Коши (схемы Адамса переменного порядка, использующие переменный шаг)

d02ejc (nag_ode_ivp_bdf_gen) - решение задачи Коши для жёсткой системы уравнений (схемы BDF)

d02gac (nag_ode_bvp_fd_nonlin_fixedbc) - решение нелинейной системы дифференциальных уравнений с заданными значениями в двух точках методом конечных разностей

d02gbc (nag_ode_bvp_fd_lin_gen) - решение линейной системы дифференциальных уравнений с заданными значениями в двух точках методом конечных разностей

d02pcc (nag_ode_ivp_rk_range) - решение задачи Коши (схемы Рунге-Кутты)

d02pzc (nag_ode_ivp_rk_errass) - оценка глобальной ошибки приближённого решения, вычисленного по схеме Рунге-Кутты

d02rac (nag_ode_bvp_fd_nonlin_gen) - решение нелинейной системы дифференциальных уравнений с граничными условиями общего вида, заданными в двух точках, методом конечных разностей

Интерполяция и аппроксимация функций

e01bac (nag_1d_spline_interpolant) - построение интерполяционной кубической сплайн функции для одномерного набора данных

e01bec (nag_monotonic_interpolant) - построение кусочно-кубического интерполяционного полинома Эрмита

e01bfc (nag_monotonic_evaluate) - вычисление значения кусочно-кубического интерполяционного полинома Эрмита

e01bgc (nag_monotonic_deriv) - вычисление значения и производной кусочно-кубического интерполяционного полинома Эрмита

e01bhc (nag_monotonic_intg) - вычисление интеграла от кусочно-кубического интерполяционного полинома Эрмита

e01dac (nag_2d_spline_interpolant) - построение интерполяционной бикубической сплайн функции для двумерного набора данных

e01sac (nag_2d_scatter_interpolant) - построение двумерной интерполяционной функции для двумерного набора данных модифицированным методом Шепарда

e02adc (nag_1d_cheb_fit) - построение ряда по полиномам Чебышева, аппроксимирующего произвольный одномерный набор данных методом наименьших квадратов

e02bac (nag_1d_spline_fit_knots) - построение кубической сплайн функции, аппроксимирующей произвольный одномерный набор данных методом наименьших квадратов

e02bbc (nag_1d_spline_evaluate) - вычисление значения кубической сплайн функции

e02bcc (nag_1d_spline_deriv) - вычисление значения и производной кубической сплайн функции

e02bdc (nag_1d_spline_intg) - вычисление интеграла от кубической сплайн функции

e02dcc (nag_2d_spline_fit_grid) - построение бикубической сплайн функции, аппроксимирующей двумерный набор данных, определённый в узлах прямоугольной сетки на плоскости (метод наименьших квадратов)

e02ddc (nag_2d_spline_fit_scatter) - построение бикубической сплайн функции, аппроксимирующей произвольный двумерный набор данных методом наименьших квадратов

Вычислительные функции среды MATLAB®

MATLAB – это высокопроизводительный язык для научных расчётов. Он включает в себя вычисления, визуализацию и программирование в удобной среде, где задачи и решения выражаются в форме, близкой к математической. Типичное использование языка MATLAB - это:

- математические вычисления
- создание алгоритмов
- моделирование
- анализ данных и визуализация
- научная и инженерная графика
- разработка приложений, включая создание графического интерфейса

В дополнение, MATLAB позволяет использовать программы написанные на языках C, C++ и Fortran.

Решение систем линейных уравнений

cond - вычисление числа обусловленности

chol - вычисление разложения Холецкого симметричной положительно определённой матрицы

lu - вычисление LU разложения

minres - решение систем линейных уравнений с разреженной симметричной матрицей итерационным методом минимальных невязок

bicg, bicgstab, cgs - решение систем линейных уравнений с разреженной матрицей итерационным методом сопряжённых градиентов
cholinc - вычисление неполного разложения Холецкого
gmres - решение систем линейных уравнений с разреженной матрицей обобщённым методом минимальных невязок
luinc - вычисление неполного LU разложения
pcg - решение систем линейных уравнений с разреженной матрицей методом сопряжённых градиентов с предобуславливанием

Задача на собственное значение

qr - вычисление QR разложения
balance – балансировка матрицы для повышения точности вычисления собственных значений
eig - вычисление собственных значений и собственных векторов матрицы
hess – вычисление формы Хессенберга

Решение нелинейных уравнений

fzero – вычисление корня одномерного уравнения (комбинация метода деления отрезка пополам и метода секущих)
roots – вычисление корней полинома

Численное интегрирование

quad - вычисление одномерного интеграла методом Симпсона
quadl - вычисление одномерного интеграла методом Лобатто
dblquad - вычисление двумерного интеграла

Решение обыкновенных дифференциальных уравнений

bvp4c - решение нелинейной системы дифференциальных уравнений с заданными значениями в двух точках методом конечных разностей
bvpinit – вычисление начального приближения для функции **bvp4c**
ode45 - решение задачи Коши для системы уравнений (явные схемы Рунге-Кутты порядка 4(5))
ode23 - решение задачи Коши для системы уравнений (явные схемы Рунге-Кутты порядка 2(3))

odel13 - решение задачи Коши для системы уравнений (схемы предиктор-корректор переменного порядка)

ode15s - решение задачи Коши для жёсткой системы уравнений (схемы BDF)

ode23s - решение задачи Коши для жёсткой системы уравнений (схема Розенброка второго порядка)

ode23t - решение задачи Коши для жёсткой системы уравнений (неявные схема Рунге-Кутты второго порядка)

ode23tb - решение задачи Коши для жёсткой системы уравнений (комбинация неявной схемы Рунге-Кутты второго порядка и схемы BDF второго порядка)

Интерполяция и аппроксимация функций

interp1 – интерполяция одномерного набора данных

interp2 - интерполяция двумерного набора данных

interp3 - интерполяция трёхмерного набора данных

interpft - интерполяция одномерного набора данных с использованием быстрого преобразования Фурье

interpn - интерполяция многомерного набора данных

Литература

Абаффи Дж., Сиендикато Э. Математические методы для решения линейных и нелинейных уравнений: Проекционные ABS-алгоритмы - М.: Мир, 1996

Абрамовиц М. Справочник по специальным функциям с формулами, графиками и математическими таблицами - М.: Наука, 1979

Батчер Дж. К., Лэмберт Дж. Л. и Протеро А. Современные численные методы решения обыкновенных дифференциальных уравнений - М.: Мир, 1979

Воеводин В. В. и Кузнецов Ю. А. Матрицы и вычисления - М.: Наука, 1984

Годунов С. К. и Рябенский В. С. Разностные схемы - М.: Наука, 1976

Годунов С. К. Современные аспекты линейной алгебры - Новосибирск: Научная книга, 1997

Голуб Дж. и Ван Лоун Ч. Матричные вычисления - М.: Мир, 1999

Икрамов Х. Д. Численное решение матричных уравнений: Ортогональные методы - М.: Наука, 1984

Калиткин Н. Н. Численные методы - М.: Наука, 1975

Лившиц К. И. Сглаживание экспериментальных данных сплайнами - Томск: Издательство Томского университета, 1991

Ортега Дж. и Пул У. Введение в численные методы решения дифференциальных уравнений - М.: Наука, 1986

Парлетт Б. Симметричная проблема собственных значений - М.: Мир, 1983

Писсанецки С. Технология разреженных матриц - М.: Мир, 1988

Райс Дж. Матричные вычисления и математическое обеспечение - М.: Мир, 1984

Самарский А. А. Введение в численные методы - М.: Наука, 1987

Самарский А. А. и Гулин А. В. Численные методы - М.: Наука, 1989

Стечкин С. Б. и Субботин Ю. Н. Сплаины в вычислительной математике - М.: Наука, 1976

Трауб Дж. Ф. Итерационные методы решения уравнений - М.: Мир, 1985

Уилкинсон Дж. Алгебраическая проблема собственных значений - М.: Наука, 1970

Фадеев Д. К. и Фадеева Н. Н. Вычислительные методы линейной алгебры – М.: Физматгиз, 1960

Хайрер Э., Нёрсетт С. и Ваннер Т. Решение обыкновенных дифференциальных уравнений: Нежёсткие задачи – М.: Мир, 1990

Хейгеман А. и Янг Д. Прикладные итерационные методы – М.: Мир, 1986

Холл Дж. и Уатт Дж. (Ред.) Современные численные методы решения обыкновенных дифференциальных уравнений – М.: Мир, 1979

Предметный указатель

А

Аппроксимация (свойство
разностной схемы) 152

Адамса схема

- явная 172

- неявная 175

Б

Бучера (Butcher) таблица 167

В

Веса квадратурной
формулы 127

Весовая функция 126

Вложенная пара 208

Г

Гаусса-Кристоффеля
формулы 126

Гершгорина круг 79

Горнера схема 219

Д

Демпфирование 110

Дифференцирования
назад формулы 188

Ж

Жёсткая система
уравнений 187

Жёсткости коэффициент 187

З

Задача на собственное значение

- - - стандартная 78

- - - обобщённая 78

И

Интеграл несобственный 132

Интерполяционный полином

- - Лагранжа 216

- - Ньютона 219

- - тригонометрический 220

Интерполяционная сплайн
функция 224

Итерационный процесс 18, 21,
57

Итерационная функция 21, 95

Итерация

- одноточечная 23, 25

- многоточечная 24, 33

- одноточечная с памятью 24,
35

К

Критерий завершения итера-
ционного процесса 25,
57

Круг Гершгорина 79

М

Матрица

- с диагональным преобла-
данием 47

- ленточная 44

- ортогональная 46

- разложимая 46

- разреженная 44

- симметричная положительно
определённая 45

- треугольная 45

- Эрмитова 78

- Хессенберга 89

- Якоби 97, 101, 162
- Машинное число 12
- Машинная арифметика 14
- Метод
 - Гаусса-Зейделя 62
 - деления отрезка пополам 19
 - с кубической сходимостью 31, 32, 33, 105
 - минимальных невязок 70
 - наименьших квадратов 232
 - наискорейшего спуска 72
 - Ньютона 30, 101
 - обратной итерации 83
 - последовательных приближений 197
 - прогонки 53
 - простой итерации 26, 95
 - последовательной верхней релаксации (SOR) 67
 - прямоугольников 115
 - QR разложения 88
 - релаксации 65
 - секущих 35
 - Симпсона 117
 - стрельбы 190
 - сопряжённых градиентов 72
 - трапеций 116
 - установления 200
 - фиктивных областей 205
 - Харди 228
 - Якоби 60

Н

- Невилла схема 216
- Невязки вектор
 - для системы линейных уравнений 55
 - для стандартной задачи на собственное значение 79
- Невязка разностной схемы 151
- Норма
 - вектора 41
 - матрицы 42

О

- Обусловленность 48
- Оператор перехода 163
- Ортогональная функция 238
- Отношение Рэлея 79
- Ошибка
 - алгоритма 48
 - возмущения 48
 - округления 48
- Ошибка квадратурной формулы 120
- Ошибка выполнения арифметических операций 15

П

- Параметр релаксации 65
 - оптимальный 66, 67
- Полином ортогональный 128
- Порядок
 - квадратурной формулы 120
 - разностной схемы 152
- Правило перемножения 135
- Предиктор-корректор схема 176
- Предобуславливание 73
- Представление числа с плавающей точкой 11
- Прямые методы 52

Р

- Разложение
 - на треугольные матрицы (LU) 52
 - Холесского 52
 - Ортогональное (QR) 52, 88
- Разностная схема 147
- Разность
 - вперёд 157
 - назад 157
 - центральная 158
- Римана интеграл 114
- Рунге-Кутта схема 167
- Рэлея отношения 79

С

- Сдвиг начала 85
- Сетка разностная 146, 150
- Сеточная функция 147, 150
- Симпсона метод 117
- Спектральный радиус 59
- Сплайн кубический 225
- Степенной метод 80
- Сходимость
 - итерационного процесса 21, 59
 - метода простой итерации 25, 96
 - метода Ньютона 102
 - метода QR 90
 - разностной схемы 159

У

- Узлы
 - интерполяции оптимальные 244
 - квадратурной формулы 127
- Устойчивость
 - метода прогонки 55
 - разностной схемы 159
 - решения задачи Коши 161

Ф

- Формулы Гаусса-Кристоффеля 126

Х

- Хессенберга форма 89
- Холесского разложение 52

Ч

- Чебышева полиномы 239
 - - нормированные 244
- Число обусловленности 49

Э

- Эйлера схема 166
- Эйткена процесс 28
- Экстраполяция по Ричардсону 122